ISSN 0860-1941

CENA 300 zł

mikroklan

vademecum techniki mikrokomputerowej 2'87

Mikroprocesory serii MC68000

Czemu wolę MOTOROLĘ

Atari 520 ST

Drukowanie barwne

Kursy języków BASIC i Pascal

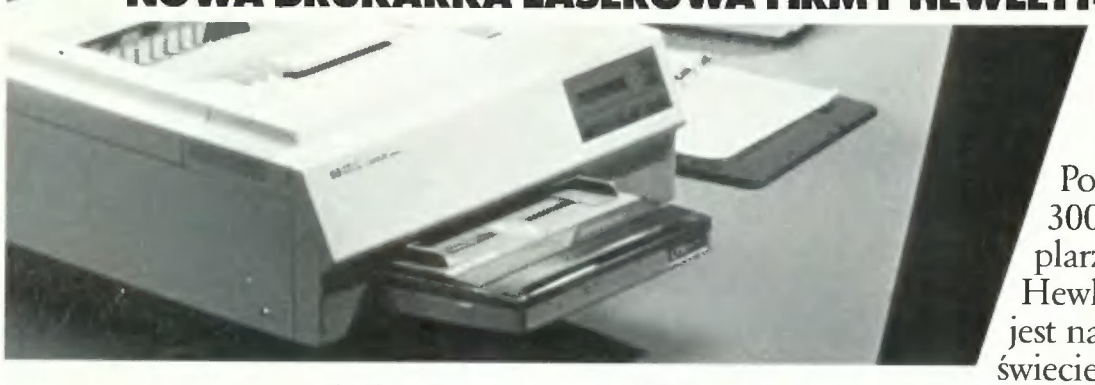
Rozbudowa Spectrum

ANKIETA i KUPON
str. 21 i IV okładka

68000

WYDAWNICTWO NOT SIGMA

LEPSZE PARAMETRY – NIŻSZA CENA
NOWA DRUKARKA LASEROWA FIRMY HEWLETT-PACKARD!



Po sprzedaniu 300.000 egzemplarzy, firma Hewlett-Packard jest największym na świecie producentem drukarek laserowych. Wprowadzając na rynek drukarkę nowej generacji – LaserJet II – firma Hewlett-Packard ustanowiła nowy standard:

● możliwość emulacji wielu różnych typów drukarek ● możliwość podłączenia do każdego komputera osobistego ● ponad 100 różnych krojów czcionek (23 realizowane sprzętowo) ● możliwość drukowania kodów paskowych (3/9 i EAN/UPC) ● obsługa przez ponad 300 gotowych programów ● pamięć wewnętrzna do 4,5 MB ● interface szeregowy i równoległy – protokół definiowany przez użytkownika ● 8 stron na minutę ● pionowe i poziome ustawienie wydruku ● możliwość drukowania na papierze, kopertach, nalepkach i folii ● cicha, bezszmerowa praca.

Zapraszamy do odwiedzenia naszego stoiska na LIX Międzynarodowych Targach Poznańskich (14–21 czerwca 1987), pawilon 2 (balkon).

Informacji udzielają:

Hewlett-Packard, Wiedeń, Lieblgasse 1, A-1222 Vienna, Austria, tel.: 00 43/02 22/25 00-0, teleks: 134425

Zotpan Warszawa, Newelska 6, tel.: 36 47 11, 37 91 44, teleks: 813919

cena: 4185 $ USA

WE'RE WITH YOU ALL THE WAY. HEWLETT PACKARD

E0/462/87

mikroklan 2 '87

Wprowadzenie
w zagadnienia
techniki
mikrokomputerowej

Redaguje zespół
miesięcznika INFORMATYKA

Maciej Adamczyk, Piotr Breitkopf,
Teresa Jabłońska (sekretarz redakcji),
Władysław Klepacz (redaktor naczelny),
Krzysztof Kontek, Jerzy Orkiszewski,
Piotr Parlewicz,
Maria Pawlak (sekretarz redakcji),
Zbigniew Pojmański, Danuta Sot,
Krzysztof Rzymkowski,
Romuald Szuniewicz
Józef Kaczmarczyk (opracowanie graficzne)

WYDAWNICTWO CZASOPISM I KSIĄŻEK TECHNICZNYCH

SIGMA

PRZEDSIĘBIORSTWO NACZELNEJ ORGANIZACJI TECHNICZNEJ

00-950 Warszawa, skrytka 1004
***ul. Biała** 4*

ISSN 0860-1941
Wydawnictwo Czasopism
i Książek Technicznych
NOT-SIGMA
Warszawa 1987

Drukowano na zlecenie ARS POLONA

Redakcja: 01-517 Warszawa,
ul. Mickiewicza 18 m. 17, tel. 39 14 34
Skład: technika fotoskładu Eurocat 150
Wydawnictwo NOT-Sigma
Druk: Bohmann Druck und Verlag
GmbH&Co. KG, Wiedeń. Austria

Nakład 100 000 egz. K-67
Cena 300 zł

SPIS TREŚCI

INFORMATYCZNE ABC

- Uczmy się Pascala! (2)
  micro 7
- BASIC dla początkujących (5)
  micro 27

OPIS

- Mikroprocesory serii MC68000
  Piotr Parlewicz 2
- Porównanie architektury 16-bitowych mikroprocesorów firm INTEL I MOTOROLA, czyli czemu wolę MOTOROLĘ
  Adam Forycki 11
- Czy przewaga technologiczna wystarczy?
  micro 14
- Drukowanie barwne
  micro 16

ZASTOSOWANIA

- PEPSY – pakiet programowy do obliczeń statycznych
  micro 4
- Tablica kontrolna systemu TV PAL
  Wojciech Świtalski 6
- Puste biurko
  micro 19

SPRZĘT

- Rozszerzenie pamięci mikrokomputera ZX Spectrum
  Krzysztof Amborski, Dariusz A. Przygoda 24

OPROGRAMOWANIE

- Czym jest UNIX (5). Jądro
  Janusz Zalewski 22

TEST

- Jakość dyskietek (2)
  micro 30

NOWOŚCI

- Układy biologiczne 15

BIT ZA TRZY GROSZE

- Spektroskopia Spectrum
  Bartek Mank III okładka

Mikroprocesory serii MC68000

Architektura dostępnych na rynku mikroprocesorów Motorola 680xx jest obecnie jedną z najnowocześniejszych. Fakt ten wynika ze śmiałej decyzji, jaką podjęli jego projektanci w 1979 roku. Zrozumieli, że rosnące możliwości technologiczne produkcji układów scalonych pozwalają już stworzyć procesor zapewniający przyszłemu programiście wygodę użytkowania. Stworzono całkowicie nowy typ mikroprocesora, który nie jest kompatybilny z dotychczasowymi konstrukcjami, ale za to przewyższa je pod każdym innym względem. Technologia końca lat siedemdziesiątych nie pozwalała jeszcze budować na jednej płytce krzemu procesora 32-bitowego, stworzono więc namiastkę ideału – procesor, który z punktu widzenia programisty był 32-bitowy, a konstruktora – 16-bitowy.

Ówczesna technologia uniemożliwiła jednak realizację innych cech docelowego procesora 32-bitowego. Między innymi MC68000 ma możliwość mnożenia dwóch liczb tylko 16-bitowych z wynikiem 32-bitowym oraz dzielenia liczby 32-bitowej przez liczbę 16-bitową, dając 16-bitową resztę i 16-bitowy iloraz. Na marginesie należy wspomnieć, że procesor MC68020 jest już pełną implementacją docelowego modelu 32-bitowego.

Dziś cena procesora MC68000 waha się w granicach od 8 do 20 dolarów (w zależności od częstotliwości zegara).

Firma Intel przyjęła natomiast inną koncepcją rozwoju. Za najważniejsze uznano zapewnienie kompatybilności z legendarnym już Intelem 8080, dlatego każdy następny procesor tej firmy jest rozwinięciem wersji poprzedniej. Z tego powodu w procesorach Intela (w tym również nowego 80286) nadal pokutuje lista rozkazów, której koncepcja, mająca już ponad 10 lat, polegała przede wszystkim na ułatwieniu napisania krótkiego programu asemblera. Należy przypomnieć, że komputery osobiste z lat sukcesów 8080 były wyposażone w pamięci o pojemności 4,8 lub 16 KB. Płytka pamięci dynamicznej o pojemności 32 KB kosztowała wtedy ponad 500 dolarów. Rozkazy były budowane tak, aby informacja o trybie adresowania była zawarta w samym rozkazie. Dzięki temu argument mógł mieć prawie zawsze taką samą postać. Dziś, kiedy pamięć dynamiczna 256 KB kosztuje zaledwie 20 dolarów, należy dążyć do maksymalnego ułatwienia pracy programisty i użytkownika. Z tych powodów asembler Motoroli 68000 jest zupełnie nowy. Jego siła wynika z 14 trybów adresowania oraz 56 podstawowych rodzajów rozkazów. Prawie każdy rozkaz może korzystać z dowolnego trybu adresowania oraz operować na argumentach 8-, 16- lub 32-bitowych.

Kombinacja tych parametrów pozwala na uzyskanie ponad 1000 możliwych rozkazów.

REJESTRY

Procesor ma osiem 32-bitowych rejestrów danych (D0-D7) oraz siedem 32-bitowych rejestrów adresowych zwanych (A0-A6). Ósmym rejestrem adresowym jest wskaźnik stosu. Istnieją dwa wskaźniki stosu, z których jeden jest w danym momencie dostępny jako A7. Jeden z nich nazwany jest **USP** (User Stack Pointer – wskaźnik stosu użytkownika), a drugi **SSP** (Supervisor Stack Pointer – wskaźnik stosu systemu nadzorczego). Który z nich jest dostępny jako A7, zależy od stanu określonego bitu w słowie **SR** (Status Register – rejestr stanu). Rejestr stanu składa się z dwóch bajtów. Bardziej znacząca połowa słowa (bity 8-15) jest dostępna tylko w trybie pracy systemowej. Mniej znacząca część jest dostępna zawsze jako **CCR** (Condition Code Register – rejestr kodu stanu). Istnieje również 32-bitowy rejestr **PC** (Program Counter – licznik programu). Znaczenie poszczególnych bitów SR wyjaśnia tabela 1.

ROZKAZ

Rozkazy MC68000 można podzielić na następujące grupy.:

Arytmetyka:

ABCD – dodawanie w BCD z uwzględnieniem bitu X (Extend),
ADD – dodawanie binarne,
SBCD – odejmowanie w BCD z uwzględnieniem bitu X,
SUB – odejmowanie binarne,
DIVS – dzielenie ze znakiem,
DIVU – dzielenie bez znaku,
MULS – mnożenie ze znakiem,
MULU – mnożenie bez znaku,
NBCD – negacja w BCD z uwzględnieniem X,
NEG – negacja binarna,
EXT – propagacja znaku.

Logika:

OR – suma logiczna,
AND – iloczyn logiczny,
EOR – różnica symetryczna (XOR),
NOT – uzupełnienie do 1.

Sterowanie:

Bcc – przejście [1] pod warunkiem **cc** (opis **cc** – patrz tabela 2),
BRA – przejście bezwarunkowe,
BSR – przejście do podprogramu,
JMP – zawsze skok [2],
JSR – skok do podprogramu,
RTD – powrót i dealokacja wg długiego słowa na stosie, dodanie do SP stałej 16-bitowej, rozciągniętej znakowo do 32 bitów,

[1] „Przejście" jest skokiem względem PC, przesunięcie może być zapisane na 8 lub 16 bitach.
[2] „Skok" jest skokiem do adresu bezwzględnego 32-bitowego.

Tabela 1. Znaczenie bitów w rejestrze stanu SR

BIT	ZNACZENIE
0	C Carry, bit flagowy przeniesienia
1	V Overflow, bit flagowy przepełnienia w arytmetyce
2	Z Zero, bit flagowy zera
3	N Negative, bit flagowy zapalony, jeśli jest ustawiony najbardziej znaczący bit wyniku
4	X Extend, służy do realizacji arytmetyki większej precyzji; działa jak C z tą różnicą, że rozkaz MOVE nie zmienia stanu X
5-7	Nie wykorzystane
8-10	Maska przerwań (I0 do I2)
11-12	Nie wykorzystane
13	S, bit stanu systemu nadzorczego. Jeśli jest ustawiony, to dostępne są pewne dodatkowe instrukcje oraz pod A7 znajduje się SSP
14	Nie wykorzystany
15	T bit stanu TRACE, jeśli jest ustawiony, to po wykonaniu kazdej instrukcji jest generowane specjalne przerwanie, ułatwiające programową realizację pracy krokowej

RTE – powrót z obsługi przerwań,
RTR – powrót i odtworzenie kodów stanu CCR wg danych ze stosu,
RTS – powrót z podprogramu,
TRAP – skok do obsługi przerwania o podanym numerze,
TRAPV – skok do obsługi przerwania od przepełnienia (bit **V** w CCR),
LINK – konstruowanie listy na stosie (działanie: *wstawienie na stos An, An:=SP, SP:=SP+d, d* jest podaną stałą 16-bitową. W celu alokacji na stosie miejsca na zmienne lokalne, należałoby podać *d < 0*),
UNLK – zdejmowanie listowe na stosie (działanie: *SP:=An, zdejmij ze stosu do An długie słowo),*
CHK – sprawdzenie, czy zawartość **Dn** należy do przedziału od 0 do limitu, jeśli nie, to następuje TRAP instrukcji CHK,
STOP – zatrzymanie pracy procesora aż do ponownego uruchomienia jako konsekwencji ustawienia bitu TRACE, względnie przerwania programowego lub sprzętowego.

Przesunięcia:

ASL – przesunięcie arytmetyczne w lewo o n bitów,
ASR – przesunięcie arytmetyczne w prawo o n bitów,
LSL – przesunięcie logiczne w lewo o n bitów,
LSR – przesunięcie logiczne w prawo o n bitów,
ROL – obrót w lewo bez X o n bitów,
ROR – obrót w prawo bez X o n bitów,
ROXL – obrót w lewo z X o n bitów,
ROXR – obrót w prawo z X o n bitów.

Operacje na bitach:

BCHG – negacja n-tego bitu, wpisanie wyniku do Z oraz na n-ty bit w słowie, z którego go pobrano,
BCLR – negacja n-tego bitu, wpisanie wyniku do Z oraz ustawienie na 0 n-tego bitu w słowie, z którego go pobrano,
BSET – negacja n-tego bitu, wpisanie wyniku do Z oraz ustawienie na 1 n-tego bitu w słowie, z którego go pobrano,
BTST – negacja n-tego bitu, wpisanie wyniku do Z,
Scc – jeśli warunek **cc** jest spełniony, to $ff zostaje wpisane do komórki wskazywanej przez adres efektywny (w przeciwnym wypadku $00),
TAS – ustawienie **cc** na podstawie zawartości bajtu wskazywanego przez adres efektywny, a następnie ustawienie w tym bajcie bitu 7,
TST – odpowiada operacji CMP, ale w wyniku tej operacji ustawia tylko bity **cc**.

Inne:

CLR – zeruje wszystkie bity argumentu,
CMP – odejmuje od siebie argumenty, ale w wyniku ustawia jedynie bity **cc,**
EXG – zamiana zawartości dwóch dowolnych rejestrów,
LEA – ładuje do rejestru adresowego adres efektywny wyliczony,
MOVE – przepisuje między argumentami
MOVEM – przepisuje dowolny podzbiór wszystkich rejestrów do adresu efektywnego, np. MOVEM. L A0-A3/D0-D5,-(a7) wpisze na stos rejestry A0, A1, A2, A3, D0, D1, D2, D3, D4 i D5,
MOVEP – przepisanie do/z urządzenia peryferyjnego, polegające na przesyłaniu po 8 bitów – najpierw bajtu najbardziej znaczącego,
MOVEQ – szybkie załadowanie rejestru dn wartością od 0 do 255 z rozciągnięciem znaku na 32 bity,
PEA – wyliczenie adresu efektywnego i załadowanie go na stos,
SWAP – zamiana 16-bitowych połówek rejestru.

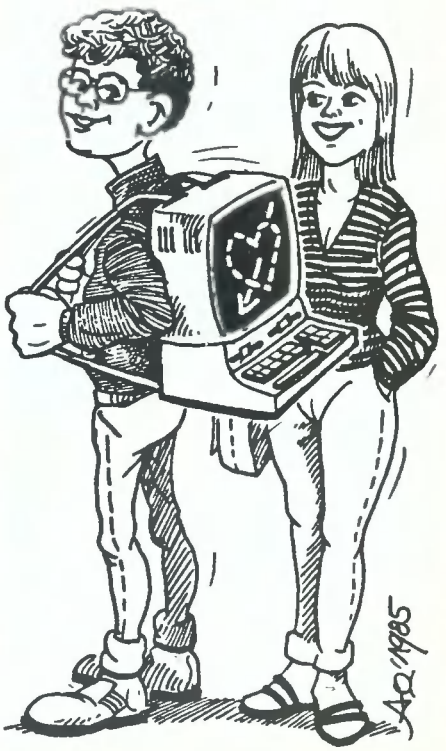
Druga część tego artykułu ukaże się w kolejnym numerze MIKROKLANU. Omówię w niej między innymi tryby adresowania oraz system przerwań MC68000.

PIOTR PARLEWICZ

Tabela 2. Działanie testów kodów stanu cc

MNEMONIK	ZNACZENIE	FUNKCJA LOGICZNA
T	true (prawda)	1
F	false (fałsz)	0
HI	high (wysoki)	/C */Z
LS	low or same (niski lub taki sam)	C + Z
CC(HS)	carry clear (przeniesienie wyzerowane)	/C
CS(LD)	carry set (przeniesienie ustawione)	C
NE	not equal (nierówne)	/Z
EQ	equal (równe)	Z
VC	overflow clear (nadmiar wyzerowany)	/V
VS	overflow set (nadmiar ustawiony)	V
PL	plus	/N
MI	minus	N
GE	greater or egual (większe lub równe)	N*V + /N * V
LT	less than (mniejsze niż)	N*/V +/N * V
GT	greater than (większe niż)	N*V*/Z + N*/V */Z
LE	less or equal (mniejsze lub równe)	Z + N*/V + /N * V

Uwaga. Znak / przed zmienną logiczną oznacza negację wartości tej zmiennej.



PEPSY – pakiet programowy do obliczeń statycznych

Gdy na przełomie wieków w Wuppertalu (RFN) wybudowano kolejkę wiszącą, mieszkańcy przesiedli się z koni do nowego środka komunikacji. W ostatnich latach unowocześnieniu uległy również środki projektowe, wykorzystywane przez inżynierów budowlanych do konstruowania takich obiektów. Dawniejsze metody wykreślne zostały z powodzeniem zastąpione przez obliczenia wykonywane za pomocą mikrokomputerów. Daje to wprost nieograniczone możliwości konstruowania przestrzennych struktur nośnych. Pokażemy, jak za pomocą mikrokomputera można przeprowadzać na ekranie złożone obliczenia konstrukcji statycznych. Jako przykład posłuży nam właśnie kolejka wisząca w Wuppertalu.

Najnowsze kierunki rozwojowe w przemyśle komputerowym sprawiły, że wielkie ośrodki obliczeniowe nie są już miejscem wykonywania bardzo złożonych obliczeń statycznych. Zadanie to przejmują mikrokomputery zainstalowane na biurkach inżynierów. Moc przetwarzania współczesnych mikrokomputerów i ich bezpośrednia dostępność zaoszczędzają inżynierom-statykom wiele czasu i pieniędzy.

Sprawdzanie konstrukcji nośnych na ekranie

Przykładem systemu programowego do obliczeń statycznych na mikrokomputerach może być program PEPSY (Pascal Engineering Programming System), opracowany przez Beyer-Hartmann-Ingenieursozietät w Erkrath-Hochdahl i przeznaczony do obliczenia konstrukcji nośnych za pomocą metody elementów skończonych. Teoretyczną podstawą programu PEPSY jest metoda przemieszczeń. Jądro stanowi pakiet rozwiązujący układy równań algebraicznych metodą Gaussa, dopasowany za pomocą najnowocześniejszych technik programowania do zastosowania w mikrokomputerach. Jako słowa kluczowe należy w tym miejscu wymienić:

- metodę frontalną,
- składanie i eliminację globalnej macierzy sztywności w miarę obliczania macierzy elementowych,
- uwzględnienie jedynie elementów niezerowych w macierzy sztywności.

Program stara się zajmować możliwie jak najmniej pamięci, dzięki czemu nawet przy dużych projektach nie brakuje pamięci operacyjnej. Jedynym ograniczeniem wielkości projektowanej konstrukcji jest pojemność dyskietki lub twardego dysku będącego do dyspozycji pamięci masowej.

Pakiet może być stosowany do obliczania wytrzymałości liniowych, sprężystych, przestrzennych konstrukcji prętowych, do projektowania stalowych instalacji przemysłowych i górniczych. Przykładem może być podpora konstrukcji nośnej kolejki wuppertalskiej.

W ramach bieżącej konserwacji postanowiono wymienić zużyte łożyska w jednej z podpór. Należało w tym celu poluzować łożyska i podnieść odpowiednią nogę podpory. Obliczenia statyczne umożliwiły wyznaczenie naprężeń, które pojawiły się w podporze na skutek przemieszczenia punktów podparcia.

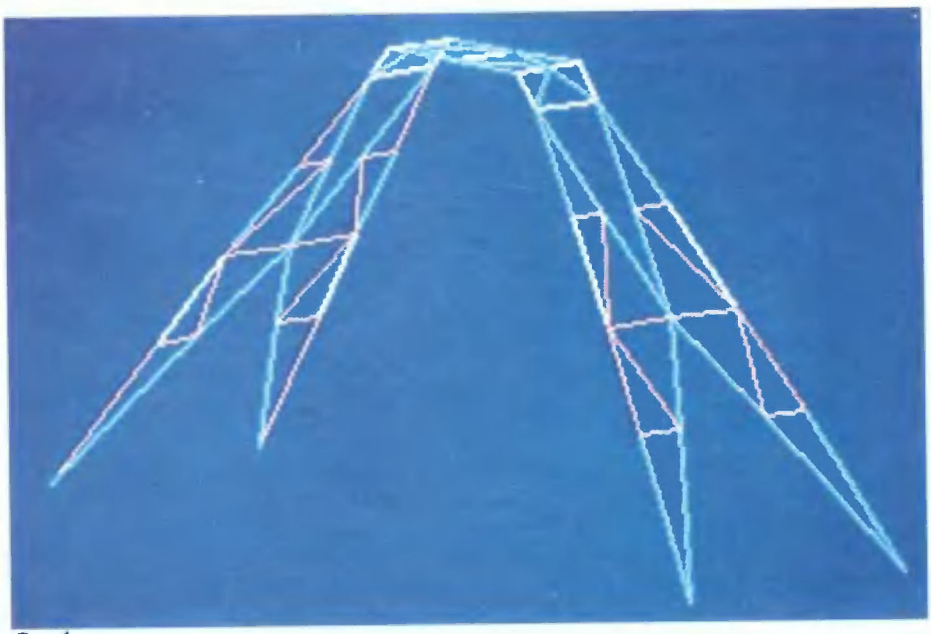
Wprowadzenie danych przez maski ekranowe

Dane do obliczeń wprowadzone są po ustaleniu modelu statycznego i jego obciążeń. Służy do tego dialog użytkownika z komputerem. Komputer wyświetla odpowiednie maski, czyli formularze, które wypełnia operator. Unika się dzięki temu wielu błędów wprowadzania, które były zmartwieniem użytkowników dużych komputerów (zredukowanie do minimum przebiegów sprawdzających).

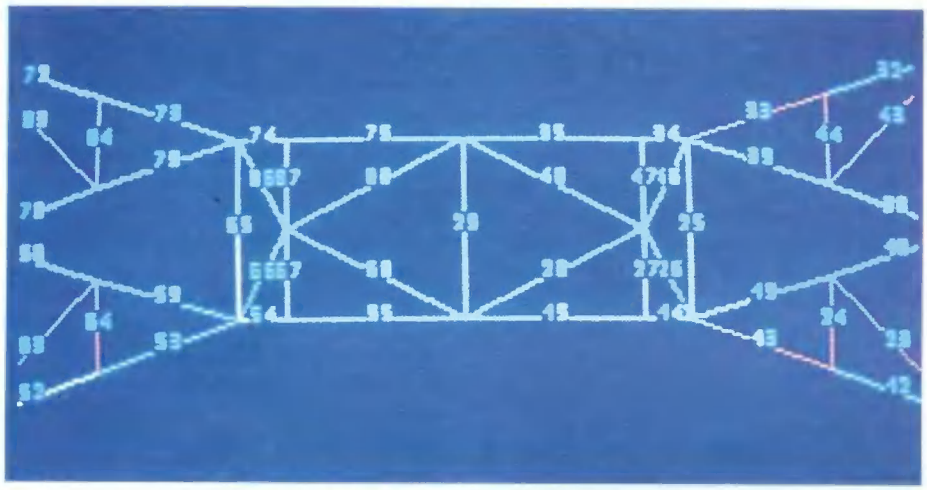
Kolejną zaletą masek ekranowych jest możliwość korzystania z informacji zawartych w podręczniku wyświetlanym na ekranie. Najważniejsze dla użytkownika wskazówki, dotyczące sposobu pracy z programem, pojawiają się na ekranie. Informacje,

Wielkości charakteryzujące złożoność przykładowych konstrukcji

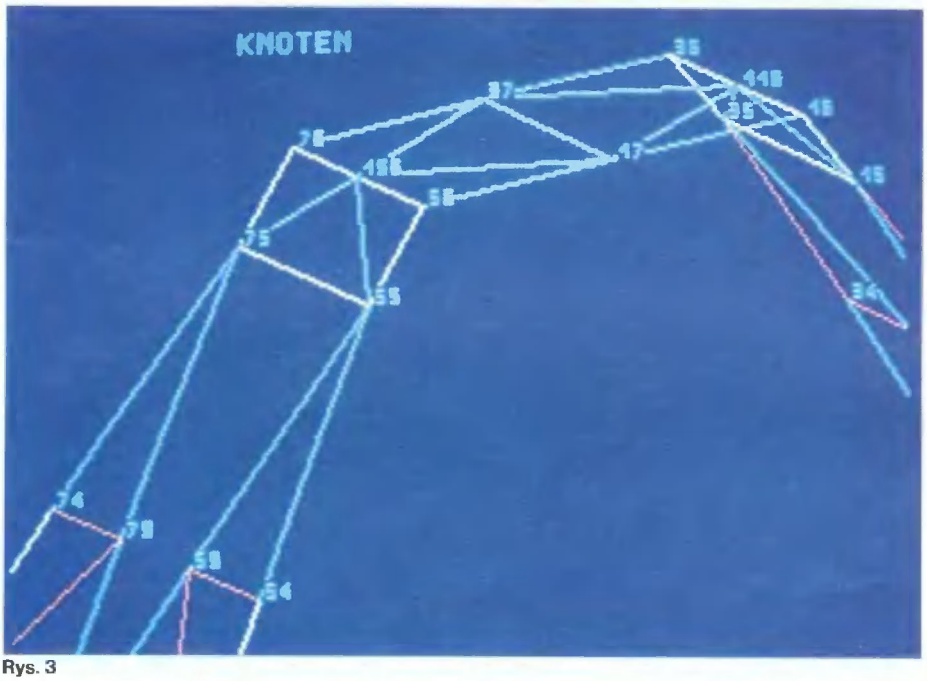
Oznaczenie	System	wz	el	po	wb	ss
A	rurociąg w elektrowni	54	64	11	36	324
B	most kratownicowy	30	71	6	30	180
C	most kratownicowy	29	38	6	30	174
D	rurociąg ciepłowniczy	37	36	2	12	222
E	żuraw budowlany	3	2	2	12	18



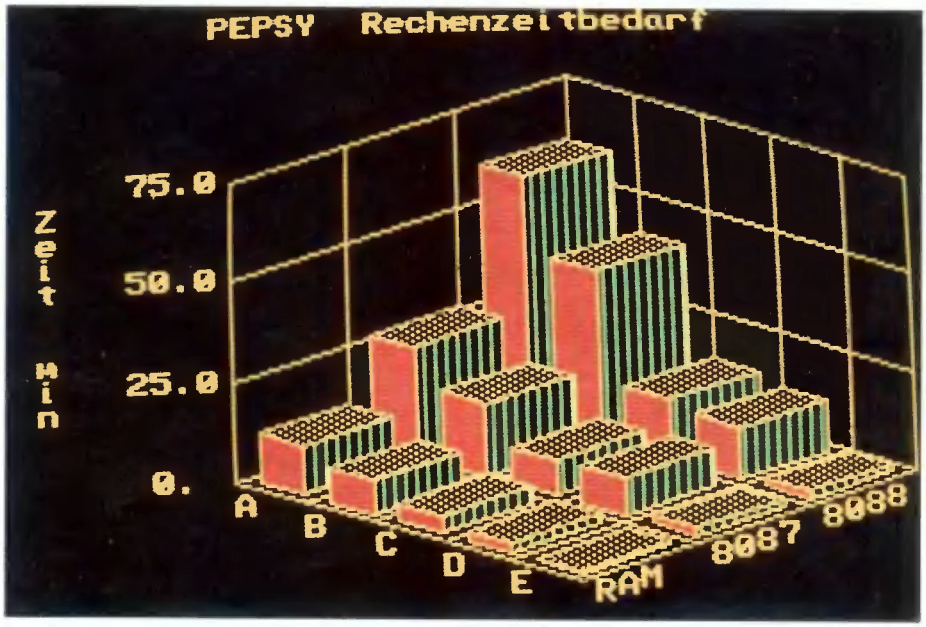
Rys. 1



Rys. 2



Rys. 3



Rys. 4

które nie są znane w momencie formułowania zadania, można zaznaczać przez wprowadzanie znaku zapytania.

Po włączeniu mikrokomputera następuje automatyczne uruchomienie programu PEPSY. Na ekranie pojawia się menu główne. Każdą z możliwości (np. **KREŚLENIE, OBLICZENIA**) wywołuje się przesuwając odpowiednio kursor lub wprowadzając cyfrę identyfikacyjną. Jeśli komputer spodziewa się podania pewnych danych, na ekranie pojawia się odpowiednia maska do ich wprowadzenia. Przy pracy z programem można używać funkcji sterujących kursorem (przesunięcie w lewo, w prawo, do góry i w dół) oraz funkcji edycyjnych (np. wyrzucenie wiersza, wstawienie wiersza, rozsunięcie zawartości ekranu i inne).

Model statyczny wraz z obciążeniem jest opisywany za pomocą masek: **WĘZŁY, ELEMENTY, TYPY I OBCIĄŻENIA.** Kolejność wywoływania tych masek jest dowolna. Wymiary poszczególnych danych w ramach danej maski mogą być wybierane niezależnie jedne od drugich. Za pomocą polecenia **KREŚLENIE** można łatwo kontrolować prawidłowość wprowadzanej struktury.

Na rys. 1 pokazano szkic podpory wyświetlony na ekranie barwnym po wydaniu polecenia **KREŚLENIE**. Różne kolory poszczególnych belek umożliwiają określenie, z których odcinków składają się poszczególne belki. Na rys. 2 przedstawiono widzianą z góry część środkową podpory wraz z numerami belek. Za pomocą klawiszy funkcyjnych, w ramach dialogu z komputerem, można błyskawicznie wybrać inną perspektywę, a także powiększyć lub zmniejszyć obraz. Rys. 3 to powiększenie górnej części podpory wraz z numerami węzłów. Oczywiście wszystkie wykresy można wyprowadzić na drukarkę graficzną.

Jądro programu PEPSY rozwiązuje wszelkie możliwe problemy statyki konstrukcji złożonych z prętów. W tym celu korzysta się z metody elementów skończonych. Można zakładać różne warunki brzegowe (np. sztywne lub sprężyste podpory) oraz obciążenia (np. wymuszone przemieszczenia podpór, siły). Przy wprowadzaniu danych i weryfikowaniu uzyskanych wyników program oferuje inżynierowi-praktykowi jeszcze wiele dalszych udogodnień.

Oprócz komfortu wprowadzania danych i prezentacji wyników oraz możliwości obliczeniowych praktyczne znaczenie ma także szybkość obliczeń. Program PEPSY potrzebował 12 minut na wyliczenie wszystkich naprężeń w węzłach, przemieszczeń węzłów i reakcji na obciążenia dla siedmiu sche-

dokończenie na str. 32

Tablica kontrolna systemu TV PAL

Czynnością niezbędną przy kontroli jakości odbiornika telewizyjnego (lub monitora) jest ocena dokładności odtwarzania tablicy kontrolnej. Analiza obrazu pozwala sprawdzić (bez pomocy specjalistycznych przyrządów) jakość kolorów i gradację kontrastu, ostrość i rozdzielczość obrazu oraz liniowość odchylania i zbieżność wiązek elektronowych.

W Polsce tablica kontrolna jest emitowana codziennie przed rozpoczęciem programu telewizyjnego i przeznaczona jest do testowania odbiorników telewizji czarno-białej i kolorowej, pracujących w systemie SECAM.

Wykorzystując fakt generowania sygnału RGB w systemie PAL przez popularne mikrokomputery ZX Spectrum, możemy za pomocą załączonego programu tworzącego tablicę kontrolną ocenić jakość odbiorników TV i monitorów pracujących w tym systemie.

Ocenę liniowości odchylania w kierunku poziomym i pionowym wykonuje się na podstawie obrazu kratownicy. Zniekształcenia natomiast ocenia się mierząc wymiary kwadratów w różnych miejscach ekranu. Wzrost różnicy wymiarów kwadratów w kierunku pionowym lub poziomym świadczy o złej liniowości odbiornika. W wypadku braku zbieżności trzech wiązek elektronów linie kratownicy ulegają zakolorowaniu w jednym z trzech podstawowych kolorów (niebieski, czerwony, zielony). Przy dokładnej zbieżności linie kratownicy są białe.

Kontrolę występowania odbić przeprowadza się obserwując obraz pojedynczych prążków (białego i czarnego) w górnym pasie ekranu. Niedopasowanie falowe odbiornika i komputera powoduje powstanie dodatkowych prążków. Za pomocą tego samego pasma tablicy kontrolnej ocenia się występowanie smużeń i prawidłowe zestrojenie toru luminancji i chrominancji. Smużenia występują przy niewłaściwej charakterystyce przenoszenia pasma wizyjnego w zakresie średnich częstotliwości i objawiają się rozmyciem pomiędzy pasem białym i czarnym. Prawidłowość zestrojenia toru luminancji i chrominancji odbiornika objawia się czystym przejściem między kolorem żółtym i czerwonym w środkowej części górnego pasa.

Wierność odtwarzania kolorów ocenia się na podstawie gamy zawierającej następujące kolory: niebieski, czerwony, purpurowy, zielony, turkusowy i żółty.

Dolny pas zawiera te same kolory oraz dodatkowo biały i czarny. Kolejność kolorów jest dobrana w ten sposób, że luminancja poszczególnych pasów zmienia się stopniowo od minimalnej (kolor czarny) do maksymalnej (kolor biały). Fakt ten pozwala na ocenę stopnia gradacji kontrastu. Przy dobrze wyregulowanym odbiorniku i wyłączonych kolorach wszystkie stopnie gradacji powinny być rozróżniane.

Szacunkową rozdzielczość poziomą obrazu można ocenić, obserwując grupy pionowych prążków środkowego pasa. Najwyższa gęstość prążków odpowiada rozdzielczości około 350 linii (standard OIRT wynosi 300 do 500).

Pewną wadą omawianej tablicy kontrolnej jest to, że obraz generowany przez komputer Spectrum nie obejmuje całej powierzchni ekranu, podczas gdy największe zniekształcenia geometryczne obrazu występują na jego obrzeżach. Jednak możliwość dokonania oceny szeregu parametrów jakości odbioru obrazu telewizyjnego przez każdego użytkownika, bez specjalistycznego przygotowania i aparatury kontrolno-pomiarowej, potwierdza przydatność programu.

WOJCIECH ŚWITALSKI

```
  5 PAPER 0: INK 7: BORDER 6: BRIGHT 1
  6 REM KRATA*******
  7 CLEAR
 10 FOR x=15 TO 240 STEP 15
 15 IF x>30 AND x<225 THEN  GO TO 25
 20 PLOT x,0: DRAW 0,175: GO TO 30
 25 PLOT x,0: DRAW 0,35: PLOT x,140: DRAW 0,35
 30 NEXT x
 50 FOR y=5 TO 170 STEP 15
 55 IF y>35 AND y<140 THEN  GO TO 65
 60 PLOT 0,y: DRAW 255,0: GO TO 70
 65 PLOT 0,y: DRAW 30,0: PLOT 225,y: DRAW 30,0
 70 NEXT y
100 REM ROZDZIELCZOSC*******
110 FOR x=30 TO 60 STEP 10
120 FOR a=x TO x+5 STEP 1
130 PLOT a,64
140 DRAW 0,31
150 NEXT a
160 NEXT x
200 FOR x=63 TO 120 STEP 6
210 FOR a=x TO x+3 STEP 1
220 PLOT a,64
230 DRAW 0,31
240 NEXT a
250 NEXT x
260 FOR x=122 TO 180 STEP 4
270 FOR a=x TO x+2 STEP 1
280 PLOT a,64
290 DRAW 0,31
300 NEXT a
310 NEXT x
320 FOR x=181 TO 224 STEP 2
340 PLOT x,64
350 DRAW 0,31
370 NEXT x
399 REM ODBICIA,SMUZENIA ******
400 FOR x=32 TO 223 STEP 1
410 IF x=83 THEN GO TO 430
420 IF x=46 OR x>60 AND x<105 OR x>150 AND x<195 THEN GO TO 450
430 PLOT x,120: INK 7
435 IF x>120 AND x<136 THEN INK 2
440 IF x>=105 AND x<=120 OR x>=136 AND x<=150 THEN INK 6
445 DRAW 0,15
450 NEXT x
500 REM GAMA KOLOROW***
505 LET k=1: BRIGHT 0
510 FOR x=32 TO 223 STEP 1
520 INK k
530 PLOT x,96
540 IF x=64 OR x=96 OR x=128 OR x=160 OR x=192 THEN LET k=k+1
550 DRAW 0,23
600 NEXT x
650 REM GRADACJA KONTRASTU***
655 LET k=1
670 FOR x=56 TO 223 STEP 1
675 INK k
680 PLOT x,40
685 IF x=80 OR x=104 OR x=128  OR x=152 OR x=176 OR x=200 THEN
LET k=k+1
690 DRAW 0,23
700 NEXT x
```

Uczmy się PASCALA! (2)

Obliczenia arytmetyczne stanowią ważną dziedzinę zastosowań komputerów. Druga część kursu pokazuje możliwości Pascala w tej dziedzinie.

W poprzednim odcinku zademonstrowano, jak program w Pascalu wczytuje dane z klawiatury i jak je wyprowadza na ekran. Przykładom brakowało istotnego elementu programu: dane nie były ani zmieniane, ani nie były wykorzystane do jakichkolwiek manipulacji. Najprostszymi operacjami na danych są działania arytmetyczne. Wyjaśnia to następujący przykład:

```
PROGRAM oblicz-a;
VAR i1, i2, iout
BEGIN
 WRITE ('pierwsza liczba:');
 READLN (i1);
 WRITE ('druga liczba:');
 READLN (i2);
 iout:=i1+i2;
 WRITELN ('wynik:', iout);
END.
```

Program ten, w przeciwieństwie do podanych w pierwszej części kursu, wykorzystuje zmienne, a więc symbole, których wartości są nadawane oraz przekształcane w trakcie jego wykonania. Jeśli zmienne te są tego samego typu, to nie trzeba ich deklarować oddzielnie, a tylko wymienić po symbolu typu danych i oddzielić przecinkami. Zapis taki nie może być jednak dłuższy niż jeden wiersz.

Użyty w przykładzie typ danych INTEGER obejmuje wszystkie liczby całkowite od –32768 do +32767. Miejsca po przecinku są więc niedozwolone. Program zapytuje najpierw operatora o dwie liczby, które za pomocą instrukcji czytania READLN przyporządkowane zostają zmiennym **i1** oraz **i2**.

Właściwe działanie arytmetyczne znajduje się w wierszu

iout:=i1+i2;

Kolejność jest najważniejsza. Zmienna, która ma zawierać wynik (w przykładzie **iout**) znajduje się po lewej stronie działania. Kombinacja znaków := oznacza operację podstawienia. Następnie zostają podane poszczególne działania arytmetyczne. Informacje o istniejących w tej mierze możliwościach podaje tabela 2. W podanym przykładzie wynik zawarty w **iout** zostaje wyprowadzony na ekran za pomocą zwykłej instrukcji zapisania jednego wiersza (WRITELN).

Jeżeli wynik operacji arytmetycznej ma być, tak jak w przykładzie, jedynie wyprowadzony, a nie użyty w dalszym przebiegu programu, wówczas istnieje następująca możliwość skrócenia programu:

```
PROGRAM oblicz-b;
VAR i1, i2:INTEGER;
BEGIN
 WRITE ('pierwsza liczba:');
 READLN (i1);
 WRITE ('druga liczba:');
 READLN (i2);
 WRITELN ('wynik:', i1+i2);
END.
```

W związku z tym, że w dalszym ciągu programu nie jest wykorzystana wartość zmiennej iout, możemy zmienną pominąć. Jak widać, w przykładzie użyto operacji arytmetycznej bezpośrednio jako argumentu procedury WRITELN. W Pascalu jest to dozwolone, ma jednak tę wadę, że późniejsze wykorzystanie wartości sumy i1 + i2 wymaga jej ponownego obliczenia.

Jeżeli jedna z pierwotnie użytych do wykonania działania liczb nie będzie już potrzebna w dalszym przebiegu programu, to narzuca się również następujący sposób wykonania działania:

i1:=i1+i2;

Pascal dodaje do początkowej zmiennej **i1** liczbę **i2**, a wyniki zapamiętuje pod adresem zmiennej **i1**. Znika jednak poprzednia wartość **i1**.

Zmienne INTEGER można również mnożyc (symbol x), odejmować (symbol –) i dzielić (symbole DIV oraz MOD). To ostatnie działanie wymaga pewnego wyjaśnienia. Wynik dzielenia, również liczb całkowitych, może mieć miejsca po przecinku (np. 9:4=2,25). Dla zmiennej INTEGER jest to niedopuszczalne. Stąd dzielenie może być wykonywane na dwa sposoby. Za pomocą operatora DIV uzyskujemy pełne liczby całkowite, a więc wynik przed przecinkiem, natomiast MOD podaje resztę z podzielenia dwóch liczb w postaci liczby całkowitej. Jeżeli **i1** ma wartość 9, a **i2** wartość 4, wówczas **iout** przy obliczeniu **iout:=9 DIV 4** uzyskuje wartość 2, a przy **iout:=i1MODi2** – wartość 1.

Pascal umożliwia wykonywanie w jednej instrukcji podstawienia więcej niż jednej operacji arytmetycznej:

i:= i1 – i2 ★ i3 + i4 + 15

Kolejność wykonywanych operacji wynika z tzw. priorytetów. Najpierw jest wykonywane mnożenie i dzielenie, a następnie dodawanie i odejmowanie. W poniższym przykładzie, dla wartości zmiennych i1=+5, i2=+3, i3=+5, i4=–5 otrzymujemy w wyniku zero. Kolejność operacji można zmienić przez użycie nawiasów, np.:

i:= (i1 – i2) ★ (i3 + i4) + 15

Wówczas wynik będzie wynosił 15. Bierze się to stąd, że pierwszeństwo mają operacje w nawiasach.

Operacje z użyciem REAL

Jeżeli przy operacji arytmetycznej są potrzebne miejsca po przecinku, wówczas należy użyć zmiennych typu REAL. Zamiast używanych w Polsce przecinków, w USA powszechnym sposobem zapisu jest stosowanie kropki (np. 12,25 zapisuje się w postaci 12.25). Dokładność, z jaką są obliczane miejsca po przecinku, zależy od konkretnej wersji Pascala.

Następujący przykład pokazuje program obliczający długość przeciwprostokątnej trójkąta prostokątnego:

```
PROGRAM oblicz-c;
VAR r1, r2:REAL;
BEGIN
 WRITE ('przyprostokątna 1:');
 READLN (r1);
 WRITE ('przyprostokątna 2:');
 READLN (r2);
 r1:=SQRT (SQR (r1) + SQR (r2));
 WRITELN ('przeciwprostokątna
',r2:5:2);
END.
```

Program oblicza trzecie ramię trójkąta prostokątnego $a^2 = b^2 + c^2$, przyporządkowując zmiennej **r1** pierwiastek sumy kwadratów **r1** i **r2**.

Wyprowadzenie zmiennych REAL na ekran odbywa się w zapisie matematycznym z wykładnikiem i mantysą (np. 112, 256 jest przedstawione w postaci 1.12256e+2). Ten sposób zapisu jest często niepożądany. W wielu wypadkach potrzeba wyświetlić jedynie ograniczona liczbę miejsc po przecinku

oraz (jak to np. występuje w tabelach) ustawić w kolumnach miejsca dziesiętne we właściwy sposób.

Jeśli przyjrzymy się ostatniemu wierszowi programu oblicz-c, to rzuca się w oczy sposób przedstawienia wyniku (r2:5:2). Obie, oddzielone od zmiennych REAL tylko dwukropkami (nie przecinkami!) liczby mają następujące znaczenie: pierwsza podaje, że wypisana wartość liczby ma się znajdować w polu o szerokości pięciu znaków. Jeżeli liczba jest krótsza, to poprzedzona zostanie pustymi znakami (spacjami) w odpowiedniej liczbie. Metodę tę można stosować również przy wyrównywaniu tekstu do prawego marginesu. Druga liczba, także oddzielona dwukropkiem, jest jednak dopuszczalna wyłącznie przy występowaniu zmiennych REAL. Podaje ona liczbę miejsc po przecinku. Oprócz tego powoduje automatyczne doprowadzenie wyniku do postaci dziesiętnej, tj. bez wykładnika.

Jeżeli zachodzi potrzeba obcięcia części dziesiętnej, wartość jest zaokrąglana. Jeżeli liczba jest zbyt długa, by zmieścić się w zadeklarowanym polu, jest wypisywana poza prawy margines.
W następującym przykładzie:

```
WRITELN ('Test':5);
WRITELN (10.25:5:2);
WRITELN (1.1568915:5:2);
WRITELN (235.2589:5:2);
```

zapis na ekranie wyglądałby następująco:

```
begin
  writeln ('test' :5);
  writeln (10.25:5:2);
  writeln (1.1568915:5:2);
  writeln (235.2589:5:2)
end.
test
   10.25
    1.16
  235.26
```

Całość stanowi bardzo prostą i wygodną metodę tworzenia tabel. Wykasowanie na monitorze miejsc po przecinku nie ma wpływu na dokładność dalszych obliczeń.

Liczby szesnastkowe

Oprócz zapisu dziesiętnego w Pascalu jest dopuszczalny również zapis szesnastkowy (heksadecymalny). Aby go zrozumieć należy zapoznać się ze strukturami danych, na jakich działają komputery. Komputer funkcjonuje w układzie dwójkowym. Cyfra binarna jest nazywana bitem. Oba możliwe stany bitu są określone jako H i L (High-Low) albo 1 i 0. Komputer może jednocześnie

Działania arytmetyczne w PASCALU

Dla wszystkich przykładów obowiązuje następująca konwencja zapisu zmiennych:

```
VAR i, i1, i2, iout: INTEGER;
    r, r1, r2, rout: REAL;
```

Następujące instrukcje znajdują się we wszystkich wersjach Pascala i są tam w jednakowy sposób stosowane:

INSTRUKCJA	OPIS	PRZYKŁADY

+	dodanie dwóch liczb. Typ danych poszczególnych liczb powinien być INTEGER lub REAL (również łącznie). Jeżeli jedna z liczb jest typu REAL, to wynik musi być zapamiętany w zmiennej tego typu	iout:=i1+i2;
		rout:=r+i;
		rout:=r1+r2;

–	odjęcie dwóch liczb. Dla typu danych obowiązują wyjaśnienia podane przy dodawaniu	iout:=i1-i2;
		rout:=r-i;
		rout:=r1-r2;

x	mnożenie dwóch liczb. Dla typu danych obowiązują wyjaśnienia podane przy dodawaniu	iout:=i1xi2;
		rout:=rxi;
		rout:=r1xr2;

/	dzielenie dwóch liczb. Powinny być typu danych REAL lub INTEGER (również łącznie). Wynik należy zawsze zapamiętywać w zmiennej REAL. Dzielenie przez zero powoduje błąd	iout:=i1/i2;
		rout:=r/i;
		rout:=r1/r2;

ABS	wartość bezwzględna liczby, co oznacza, że znak ujemny liczby zostaje zmieniony na znak dodatni. Zgodnie z tym ABS (-10) daje wynik 10. Typy danych powinny być REAL lub INTEGER; wynik odpowiada pierwotnemu typowi danych	iout:=ABS(i);
DIV	wynik dzielenia zaokrąglony do liczb całkowitych. Może być stosowany jedynie typ INTEGER	iout:=i1 DIV i2;
MOD	reszta dzielenia w liczbach całkowitych. Typy danych jak przy DIV	iout:=i1 MOD i2;
ROUND	zaokrągla zmienną REAL od wartości 0,5 po przecinku w górę, poniżej tej wartości – w dół. Wynikiem jest zawsze zmienna INTEGER	iout:=ROUND(r);
SQR	kwadrat liczby. Zmienna może być typu REAL lub INTEGER. Wynik musi być zapamiętany w tym samym typie danych	iout:=SQR(i); rout:=SQR(r);
TRUNC	wartosc po odrzuceniu części dziesiętnej	iout:=TRUNC(r);

Tabela 1

W poniższych instrukcjach liczba początkowa może być typu INTEGER lub REAL, natomiast wynik jest zawsze typu REAL. W Pascalu MT+ instrukcje te można użyć tylko przy stosowaniu arytmetyki zmiennoprzecinkowej (zawsze za pomocą FPREALS, a nie BCDREALS, bezpośrednio przed PASLIB). W wersjach USCD lub Apple-Pascal bezpośrednio po nazwie programu należy umieścić wiersz

USES TRANSCEND;

Dalsze szczegóły zawarte są w podręcznikach odpowiedniej wersji języka Pascal.

INSTRUKCJA	OPIS	PRZYKŁADY
ARCTAN ATAN	arcus tangens liczby. Składnia ATAN jest stosowana tylko w wersjach USCD oraz Apple	rout:=ARCTAN(r); rout:=ARCTAN(i);
COS	cosinus liczby	rout:=COS(r); rout:=COS(i);
EXP	wartość funkcji wykładniczej liczby	rout:=EXP(r); rout:=EXP(i);
LN	logarytm naturalny liczby	rout:=LN(r); rout:=LN(i);
SIN	sinus liczby	rout:=SIN(r); rout:=SIN(i);
SQRT	pierwiastek kwadratowy liczby	rout:=SQRT(r); rout:=SQRT(i);

Poniższa grupa instrukcji może być stosowana tylko w wersji TURBO-Pascal. Liczba początkowa może być typu REAL lub INTEGER, natomiast wynik – zawsze typu REAL.

INSTRUKCJA	OPIS	PRZYKŁADY
FRAC	wartość liczby po przecinku. Jeżeli jest ona typu INTEGER, to wynikiem FRAC jest zawsze 0 jako zmienna REAL	rout:=FRAC(r);
INT	wartość liczby przed przecinkiem. Jeżeli jest ona typu INTEGER, to wynikiem INT jest zawsze wartość początkowa jako zmienna REAL	rout:=INT(r);

przetwarzać 8 lub 16 bitów (w komputerach 16-bitowych). Grupa złożona z ośmiu bitów jest nazywana bajtem. Cyfra heksadecymalna stanowi połówkę bajtu albo cztery bity. Cyfra taka jest lepiej dostosowana do wewnętrznej struktury komputera niż cyfra dziesiętna. Dlatego programista, zwłaszcza w wypadku realizacji zadań z zakresu sterowania maszyn i urządzeń peryferyjnych, będzie pracował w systemie szesnastkowym.

Cyfry szesnastkowe obejmują zakres od 0 do 9, a następnie litery od A do F. Wartości szesnastkowe, odpowiadające liczbom dziesiętnym od 0 do 15, są następujące:

DZIESIĘTNIE	SZESNASTKOWO	DWÓJKOWO
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

Liczby szesnastkowe są oznaczone w Pascalu za pomocą poprzedzającego je znaku dolara ($E odpowiada liczbie dziesiętnej 14). Za pomocą czterech bitów można utworzyć szesnaście różnych wartości. Komputery pracują na ośmiu bitach, więc jeden bajt może mieć 256 (16x16) różnych wartości (od 0 do 255 lub od $00 do $FF). W komputerach 16-bitowych liczba ta wewnętrznie jest jeszcze większa – dla programujących w Pascalu nie ma to jednak w tej chwili żadnego znaczenia. Istotne jest to, że liczby szesnastkowe nie różnią się od liczb dziesiętnych, a mają jedynie odmienną postać reprezentacji.

Dane typu znakowego CHAR są zapamiętywane w jednym bajcie. Dlatego istnieje 256 różnych znaków, przy czym znaczna ich

część jest używana jako znaki sterujące, nie dające się wyswietlić na ekranie. W praktyce wystarcza z reguły pierwszych 128 znakow. Cały ten „alfabet" nazywa się zestawem znaków ASCII (tabela 1).

Aby pracować w Pascalu tymi znakami ASCII, które nie mają swych odpowiedników w normalnym alfabecie, utworzono funkcję CHR (nie należy mylić z typem danych CHAR).

Wiersz programu:

```
WRITE (CHR (27));
```

wyprowadza na ekran znak ESCAPE (ASCII 27 lub $1B), który wielokroć jest stosowany do wprowadzania sekwencji sterujących urządzeniami peryferyjnymi.

Odwrotną funkcją w stosunku do CHR jest ORD.

```
WRITE (ORD ('A') );
```

Wyprowadza na ekran liczbę porządkową litery A, a więc liczbę 65.

Zmienne INTEGER w przeciwieństwie do zmiennych CHAR są zapamiętywane w dwóch bajtach. Zakres wartości obejmuje 65536 (256×256) różnych liczb, przy czym szesnasty bit określa znak liczby. Zakres obliczeń sięga od –32767 do +32768. W zapisie szesnastkowym zakres ten obejmuje wartości od $0000 do $FFFF, przy czym od $8000 zaczynają się liczby ujemne. Podawane niektóre podstawy teoretyczne programowania ułatwiają wprawdzie zrozumienie określonych czynności, nie są jednak nieodzowne dla usprawnienia umiejętności programowania w Pascalu. Można się całkiem obejść bez liczb szesnastkowych, ponieważ dla każdej z ich wartości istnieje odpowiednik dziesiętny. Niestety posługiwanie się ORD oraz CHR należy bezwzględnie opanować

W następnej części pokażemy, jak w Pascalu wykonuje się rozgałęzienia oraz pętle, a więc jak można sterować kolejnością wykonywania instrukcji.

micro

Tabela 2

Zestaw znaków ASCII

Znaki ujęte w nawiasach są jedynie określeniami symboli, tak jak są one np. stosowane w podręcznikach urządzań peryferyjnych. Tabela obejmuja znaki ASCII od 0 do 127. W znakach od 128 do 255 należy dodawać do wartości dziesiętnej 128 oraz wartości szesnastkowej $80. Liczby szesnastkowe są zapisywane w konwancji powszechnie stosowanej w Pascalu.

Dziesiętnie	Szesnastkowo	Znak	Dziesiętnie	Szesnastkowo	Znak	Dziesiętnie	Szesnastkowo	Znak	Dziesiętnie	Szesnastkowo	Znak
0	$00	(^a NUL)	32	$20	(SPC)	64	$40	a §	96	$60	
1	$01	(^A SOH)	33	$21	!	65	$41	A	97	$61	a
2	$02	(^B STX)	34	$22	"	66	$42	B	98	$62	b
3	$03	(^C ETX)	35	$23	#	67	$43	C	99	$63	c
4	$04	(^D EOT)	36	$24	$	68	$44	D	100	$64	d
5	$05	(^E ENQ)	37	$25	%	69	$45	E	101	$65	e
6	$06	(^F ACK)	38	$26	&	70	$46	F	102	$66	f
7	$07	(^G BEL)	39	$27	'	71	$47	G	103	$67	g
8	$08	(^H BS)	40	$28	(	72	$48	H	104	$68	h
9	$09	(^I HT)	41	$29	)	73	$49	I	105	$69	i
10	$0A	(^J LF)	42	$2A	*	74	$4A	J	106	$6A	j
11	$0B	(^K VT)	43	$2B	+	75	$4B	K	107	$6B	k
12	$0C	(^L FF)	44	$2C	,	76	$4C	L	108	$6C	l
13	$0D	(^M CR)	45	$2D	-	77	$4D	M	109	$6D	m
14	$0E	(^N SO)	46	$2E	.	78	$4E	N	110	$6E	n
15	$0F	(^O SI)	47	$2F	/	79	$4F	O	111	$6F	o
16	$10	(^P DLE	48	$30	0	80	$50	P	112	$70	p
17	$11	(^Q DC1)	49	$31	1	81	$51	Q	113	$71	q
18	$12	(^R DC2)	50	$32	2	82	$52	R	114	$72	r
19	$13	(^S DC3)	51	$33	3	83	$53	S	115	$73	s
20	$14	(^T DC4)	52	$34	4	84	$54	T	116	$74	t
21	$15	(^U NAK)	53	$35	5	85	$55	U	117	$75	u
22	$16	(^V SYN)	54	$36	6	86	$56	V	118	$76	v
23	$17	(^W ETB)	55	$37	7	87	$57	W	119	$77	w
24	$18	(^X CAN)	56	$38	8	88	$58	X	120	$78	x
25	$19	(^Y EM)	57	$39	9	89	$59	Y	121	$79	y
26	$1A	(^Z SUB)	58	$3A	:	90	$5A	Z	122	$7A	z
27	$1B	(^[ESC)	59	$3B	;	91	$5B	[Ä	123	$7B	{ ä
28	$1C	(^\ FS)	60	$3C	<	92	$5C	\ Ö	124	$7C	| ö
29	$1D	(^] GS)	61	$3D	=	93	$5D	] Ü	125	$7D	} ü
30	$1E	(^^ RS)	62	$3E	>	94	$5E	^	126	$7E	~ ß
31	$1F	(^_ US)	63	$3F	?	95	$5F	_	127	$7F	(DEL)

Powtórzenie najważniejszych pojęć

BEGIN – początek właściwego programu (części instrukcyjnej)

binarny – dwójkowy

bajt – grupa ośmiu bitów

CHAR – typ danych, obejmujący znaki pisarskie, takie jak litery, cyfry i znaki przestankowe

CHR – funkcja języka Pascal, realizująca znak pisarski typu CHAR zgodnie z numerem kodu ASCII

END. – koniec programu (uwaga na kropkę)

ESCAPE – nazwa znaku nr 27 w kodzie ASCII

szesnastkowy – system liczbowy oparty na podstawie 16

INTEGER – typ danych: liczba całkowita w zakresie od -32767 do +32768

ORD – funkcje języka Pascal, realizujące numer kodu ASCII zmiennej typu CHAR

READ, READLN – instrukcje wprowadzenia tekstu za pomocą klawiatury

REAL – typ danych: liczba z cyframi po przecinku (np. 12.2547)

VAR – początek części programu zawierającej definicje zmiennych

zmienna – dowolnie wybrana nazwa reprezentująca pojęcie, którego podstawowe znaczenie oraz typ powinnybyć wcześniej zdefiniowane, jednak bez dokładnej wartosci

WRITE, WRITELN – instrukcja wyprowadzenia tekstu na ekran. WRITELN ustawia dodatkowo kursor na początku następnego wiersza

Porównanie architektury 16-bitowych mikroprocesorów firm INTEL i MOTOROLA czyli

czemu wolę MOTOROLĘ

Styk pomiędzy komputerem a jego oprogramowaniem jest określony przez architekturę procesora. Definiuje ona zbiór dostępnych rejestrów, model pamięci, elementarne typy danych, tryby adresowania i listę rozkazów.

Architektura wczesnych 8-bitowych mikroprocesorów niezbyt odpowiada wymogom współczesnej inżynierii oprgramowania. Jest to wynikiem zarówno ich 8-bitowego charakteru i ówczesnych ograniczeń technologicznych, jak i faktu, że w fazie projektowej nie poświęcono wystarczająco dużo uwagi programowym aspektom architektury. Mikroprocesory te nie były pomyślane jako konkurencja dla tradycyjnych komputerów, lecz jako tania alternatywa dla coraz bardziej złożonych układów logicznych, budowanych dotychczas na obwodach średniej skali integracji.

Pojawienie się mikroprocesorów 16- i więcej bitowych otworzyło przed mikrokomputerami nowe zastosowania, dostępne dotychczas jedynie dla dużych systemów. Aby mikroprocesor mógł efektywnie sprostać także nowym dla niego, szerokim klasom zagadnień, architektura powinna optymalizować osiągi systemu bez nakładania na jego projektanta i użytkownika sztywnych ograniczeń, zawężających zastosowania, bądź czyniących je kłopotliwymi w realizacji.

Szesnastobitowe mikroprocesory dominujących na rynku firm Intel i Motorola tworzą już całą rodzinę. Dla wyboru porównywanych przedstawicieli rodzin przyjmijmy kryterium najniższej porównywalnej ceny. Obaj „ojcowie" rodzin, mikroprocesory Intel 8086 i Motorola 68000, dostępni są w USA w zbliżonej cenie około 9 dolarów (BYTE, maj 1986). Cóż oferują za to konkurujące firmy? Porównajmy.

Rodowód

Architektura mikroprocesora 8086 jest rozszerzeniem starej, 8-bitowej architektury weterana 8080 i przejęła większość jego wad. Podejście takie zostało podyktowane chęcią zapewnienia pewnego stopnia zgodności programowej „w dół", ale utrudnia dalsze utrzymanie zgodności w ramach rodziny. Architektura M68000 jest całkowicie nowa, wzorowana nie na makroprocesorach poprzedniego pokolenia, lecz raczej na nowoczesnych superminikomputerach czy nawet dużych systemach. Zaprojektowane z rozmachem i pewnym zapasem, zapewnia zgodność „w górę" i z przyszłymi zastosowaniami oraz członkami rodziny.

Rejestry

Podczas wykonywania programu najczęściej są wykorzystywane dostępne dla programisty rejestry procesora. Od ich liczby i łatwości dostępu zależy wygoda programowania oraz objętość i szybkość wykonywanego programu.

Procesor 8086 zawiera osiem 16-bitowych rejestrów „ogólnego przeznaczenia": **AX, BX, CX, DX, BP, SP, SI i DI.** Pierwsze cztery z nich mogą być wykorzystywane jako osiem 8-bitowych rejestrów. Używane przez firmę Intel określenie „rejestry ogólnego przeznaczenia" w zasadzie nie odpowiada prawdzie. Szereg rozkazów i trybów adresowania korzysta wyłącznie z określonych, przypisanych im rejestrów. I tak, na przykład, rejestr **AX** jest przypisany do rozkazów mnożenia, dzielenia, we-wy, arytmetyki ASCII i BCD, części rozkazów z argumentem bezpośrednim, rozkazów rozszerzenia znaku, odczytu wskaźników stanu procesora, konwersji, rozkazów blokowych itd. Zawartość rejestru **AX,** podobnie jak i rejestrów **CX** i **DX,** nie może być wykorzystana przy określeniu adresu argumentów. W zasadzie rejestr **AX** pełni w procesorze rolę pojedynczego akumulatora. Powoduje to jego przeciążenie i konieczność częstego zapamiętywania i odtwarzania jego zawartości. Podobnie i inne rejestry są przypisane do pewnych klas rozkazów lub trybów adresowania. Czyni to programowanie żmudnym, odciąga uwagę od istoty realizowanego algorytmu, zwiększa objętość i zwalnia wykonanie kodu wynikowego, szczególnie dla kodów generowanych przez kompilatory.

Procesor M68000, poza licznikiem rozkazów i rejestrem stanu, ma 17 dostępnych dla programisty rejestrów: 8 rejestrów danych D0-D7 oraz 9 rejestrów adresowych A0-A7 i A7'. Rejestry A7 i A7' pełnią funkcję wzajemnie niezależnych sprzętowych wskaźników stosu w dwóch możliwych trybach pracy procesora – trybie użytkownika i trybie uprzywilejowanym. Zawartość rejestrów może być traktowana jako wielkość 32-,16- lub – tylko dla rejestrów danych – 8-bitowa. W ramach grupy rejestry są całkowicie uniwersalne – żaden z rozkazów procesora nie wymaga wykorzystania dedykowanego rejestru (wyjątek stanowi sprzętowy wskaźnik stosu, adresowany implicite m.in. przez rozkazy skoku i powrotu do i z podprogramów). Ułatwia to programowanie i zmniejsza liczbę przesłań pomiędzy pamięcią a rejestrami.

Nie ma jednak róży bez kolców. Mniejsza liczba rejestrów i ich sztywne powiązanie z instrukcjami zmniejsza długość kodu rozkazów. Najkrótszy rozkaz 8086 zajmuje 1 bajt, najdłuższy – 6 bajtów. Analogiczne wielkości dla M68000 wynoszą odpowiednio 2 i 10. Pamiętajmy jednak, że te najdłuższe rozkazy M68k[1]) zawierają 32-bitowe dane i adres. Wydłużenie kodu rozkazów procesora M68000 zrekompensowane jest ich większą elastycznością i efektywnością, a także przejrzystą architekturą procesora. Ten ostatni element ma niebagatelny wpływ na szybkość przyswojenia sobie przez programistę listy rozkazów, liczbę popełnionych przy programowaniu błędów, szybkość pisania i uruchamiania programów, a także ich jakość.

Ważną cechą procesora M68000, związaną z podziałem rejestrów na dwie grupy, jest niezależność wskaźników stanu procesora od operacji na adresach. Dzięki temu operacje adresowe nie kolidują z będącymi w toku operacjami na danych.

Typy danych

Jedną z miar możliwości procesora jest rozróżniana na poziomie sprzętowym różnorodność typów danych, sposobów dostępu do nich i wykonywanych na nich operacji. Im większa różnorodność, tym bardziej uniwersalny procesor i szerszy zakres jego efektywnych zastosowań.

Procesor Motoroli oferuje więcej typów danych niż jego intelowski partner. W rodzinie M68k, rozkazy arytmetyczne, logiczne, porównywania i przesyłania operują 8-, 16- i 32-bitowymi danymi. Gdyby nie brak rozkazów mnożenia 32-bitowych czynników i dzielenia z 64-bitową dzielną, z programowego punktu widzenia M68000 można by uznać za w pełni 32-bitowy mikroprocesor. Procesor 8086 i jego bracia zamykają się w świecie 8 i 16 bitów, już przy 32 bitach wołając na pomoc swych numerycznych kuzynów, razem ze związanymi z tym narzutami finansowymi i czasowymi. Trudno zbagatelizować ten brak.

Z drugiej strony, efektywne operowanie najmniejszymi jednostkami informacji – indywidualnymi bitami – jest bardzo istotne przy programowaniu operacji we-wy, sterowaniu urządzeniami zewnętrznymi, w operacjach na znacznikach, w przydzielaniu zasobów, w grafice komputerowej itd. Rodzina M68k daje programiście do dyspozycji wiele szybkich i wygodnych rozkazów, które pozwalają bezpośrednio sprawdzać, ustawiać, kasować lub zmieniać stan dowolnego bitu, w dowolnym rejestrze procesora, urządzenia zewnętrznego lub komórce pamięci. Nie mają one swoich odpowiedników w repertuarze intelowskich procesorów.

Aby być sprawiedliwym Intel oferuje w zamian pewne udogodnienia w arytmetyce znaków ASCII. Dziękuję, nie zamieniam się.

Model pamięci

Fizyczna przestrzeń adresowa procesora 8086, o wielkości 1 MB, podzielona jest na segmenty o maksymalnej długości 64 KB. W każdym momencie procesor ma bezpośredni dostęp do czterech segmentów – jednego dla kodu programu, jednego dla stosu oraz dwóch dla danych. Zaletą takiego rozwiązania jest prostota realizacji sprzętowej, zwartość kodu (adres logiczny zajmuje tylko 16 bitów) i jego naturalna przemieszczalność.

Zalety te tracą jednak na znaczeniu przy wyjściu poza granicę 64 KB. Przekroczenie tej bariery jest związane z intensywną gimnastyką programową, odciągającą uwagę programisty od istoty realizowanego algorytmu. O istnieniu tej bariery, nie występującej w M68000, nie można zapomnieć. W profesjonalnych zastosowaniach wielkość struktur danych często przekracza 64 KB. Jest to wręcz regułą w zastosowaniach graficznych, sztucznej inteligencji, bazach danych, dużych programach obliczeniowych itp. Taniejące obwody pamięci – obecny przemysłowy standard pojemności obwodu pamięciowego to 256 KB – sprzyjają stosowaniu szybkich, pamięciochłonnych algorytmów. W tych warunkach nie tylko wielkość segmentu, ale i wielkość całej fizycznej przestrzeni adresowej procesora 8086 może stać się nieprzyjemnym ograniczeniem. Zaczyna to być widoczne chociażby na przykładzie IBM PC.

W przeciwieństwie do 8086, rodzina M68k ma wygodną dla programisty liniową przestrzeń adresową o wielkości 4 GB; w procesorze M68000 jedynie zewnętrznie ograniczoną do 16 MB. W połączeniu z dostępnością 32-bitowych typów danych stanowi to istotne ułatwienie przy przenoszeniu oprogramowania z dużych systemów, np. typu VAX, oraz zapewnia programową zgodność „w górę" z całkowicie 32-bitowym mikroprocesorem M68020. Ceną tego udogodnienia jest mniejsza zwartość kodu, ale chyba warto ją zapłacić. Zresztą nie zawsze jest to konieczne – przy dostępie do pierwszych i ostatnich 32 KB przestrzeni adresowej oraz w trybie adresowania względem licznika rozkazów można stosować krótki, 16-bitowy adres.

Tryby adresowania

Zbiór dostępnych trybów adresowania jest jedną z kluczowych cech architektury procesora. Określa on możliwości dostępu do struktur danych, na których działa program, a więc tym samym łatwość realizacji algorytmu dostępu, zwartość kodu i szybkość wykonywania programu.

W tabeli przedstawione zostały możliwe kombinacje uniwersalnie dostępnych trybów adresowania argumentów źródłowych i docelowych procesora M68000 (M), wraz z odpowiadającymi im funkcjonalnie kombinacjami, zrealizowanymi w procesorze 8086 (I). Liczba opcji określa możliwość różnych wariantów trybu, związanych z jego stosowalnością z różnymi rejestrami (w nawiasach liczba opcji dla procesora 8086).

Jak widać, procesor M68000 góruje nad swoim intelowskim odpowiednikiem. Pozwala między innymi na indeksowanie 32-bitowym adresem, co ma kolosalne znaczenie przy działaniach na dużych strukturach danych. Większa liczba rejestrów i ich uniwersalny charakter zwiększa liczbę wariantów adresowania.

Brak w procesorze 8086 trybów adresowania względnego, pozwalającego pisać programy przemieszczalne, zrekompensowany jest częściowo segmentową organizacją pamięci. Dużym niedostatkiem 8086 jest brak uniwersalnych trybów adresowania pośredniego przez rejestr z predekrementacją i postinkrementacją (w 8086 tryby te, ograniczone do zakresu 64 KB, używane są jedynie niejawnie w jednokrokowych odmianach rozkazów operacji blokowych i w rozkazach PUSH i POP). Rozpatrywane tryby adresowania są bardzo przydatne przy działaniach na strukturach stosowych lub kolejkowych, przy często spotykanych działaniach w pętli programowej na kolejnych elementach tablic. Warto dodać, że w M68000 wielkość, o którą rejestr zostaje zwiększony lub zmniejszony (1, 2 lub 4) jest automatycznie dostosowana do typu argumentu (bajt, słowo lub podwójne słowo).

Tabela nie ilustruje w pełni zastosowania trybów adresowania, tzn. liczby rozkazów, z którymi można je używać. Jednak i tutaj M68000 góruje nad 8086, dając możliwość m.in. operacji arytmetycznych pamięć-pamięć. Ważną cechą architektury M68000 jest daleko posunięta symetryczność, czyli możliwość użycia dowolnego typu danych z dowolnym rozkazem i trybem adresowania. Cechy tej nie ma procesor 8086, a ma ona istotne znaczenie dla implementacji algorytmicznych języków wysokiego poziomu generujących zwarty i szybki kod.

Tryby adresowania 8086 są efektywne jedynie w aktualnie dostępnych segmentach. Każdy dostęp do adresów znajdujących się poza nimi wymaga uprzedniego przeładowania wskaźnika segmentu. W ten sposób 8086 nie ma prawdziwie bezwzględnego trybu adresowania – dostęp do dowolnego adresu fizycznej pamięci nie może być zrealizowany w jednym rozkazie. Niedostatek ten jest w pewnym stopniu zrekompensowany przez istnienie odrębnej przestrzeni adresowej dla urządzeń we-wy, które są typowym przykładem stosowania adresów bezwzględnych. Jednakże operujące w tej przestrzeni rozkazy ograniczają się tylko do transferu danych i tylko 256 pierwszych adresów 64 KB przestrzeni we-wy jest dostępne bezpośrednio. Dostęp do pozostałej części jest możliwy jedynie pośrednio, przez rejestr **DX**.

W systemie opartym na procesorze M68000 przestrzeń we-wy odwzorowywana jest zwykle w najstarszej 32 K przestrzeni adresowej, co umożliwia stosowanie krótkiego, 16-bitowego adresu bezwzględnego. Wszystkie instrukcje arytmetyczne, logiczne, porównań, przesłań i działań na indywidualnych bitach mogą operować zawartością rejestrów urządzeń zewnętrznych.

Lista rozkazów

Różnice w listach rozkazów, bezpośrednio związane z uprzednio omawianymi aspektami architektury porównywanych procesorów, były już sygnalizowane. Przypomnijmy najbardziej istotne – brak w 8086 rozkazów operujących 32-bitowymi danymi i indywidualnymi bitami, brak w M68000 rozkazów arytmetyki ASCII, większa moc rozkazów procesora M68000 związana z bardziej rozbudowanymi trybami adresowania.

Często stosowane porównywanie długości list rozkazów mikroprocesorów znacznie różniących się architekturą pozbawione jest sensu. Brak na jednej liście bezpośrednich odpowiedników rozkazów z drugiej listy może być spowodowany bądź ich zbytecznością w danej architekturze (np. rozkazy ładowania wskaźników segmentów), bądź też faktem, że są one szczególnym wypadkiem bardziej ogólnego rozkazu. Celowe jest natomiast porównanie funkcjonalnosci rozkazów. Zwróćmy uwagę na podstawowe różnice.

W grupie rozkazów przesyłania, szereg odrębnych w 8086 rozkazów odpowiada rozkazowi MOVE procesora M68000, użytemu z odpowiednim trybem adresowania. Dotyczy to rozkazu XLAT (używanemu przy konwersji kodów) i rozkazów organizacji stosu.

Wspomnieć trzeba, że w 8086 jest tylko jeden wskaźnik stosu, a w M68000 może być do 8. Nie mają swoich odpowiedników takie rozkazy M68000 jak: MOVEM (przesyłający dowolną kombinację 16 rejestrów procesora, bardzo wygodny i efektywny przy przełączaniu kontekstu programowego) czy też rozkazy LINK i UNLINK, bardzo pomocne przy implementacji języków wysokiego poziomu.

W grupie rozkazów arytmetycznych, rozkazom INC i DEC procesora 8086, zmieniającym wartość argumentu docelowego o 1, odpowiadają w M68000 znacznie mocniejsze rozkazy ADDQ i SUBQ, umożliwiające zmianę tej wartości w zakresie od 1 do 8.

Dzięki rozbudowanym trybom adresowania procesora M68000 bardziej funkcjonalne są także jego rozkazy skoków bezwarunkowych. Mały zakres skoków warunkowych procesora 8086 (od +127 do –128 wobec ±32K w M68000) w wielu wypadkach pociąga za sobą konieczność stosowania dwurozkazowej sekwencji skoku warunkowego i skoku bezwarunkowego. Będący odpowiednikiem rozkazu LOOP, rozkaz DBcc procesora M68000 oferuje więcej warunków zakończenia wykonywania pętli i może współdziałać z dowolnym z 8 rejestrów danych procesora.

Na liście rozkazów procesora M68000 nie występują w sposób jawny instrukcje blokowe. Jednakże użycie w rozkazie trybów adresowania pośredniego przez rejestr z postinkrementacją lub predekrementacją, w połączeniu z rozkazem DBcc umożliwia stworzenie znacznie większego, niż w 8086, repertuaru operacji blokowych, włącznie z blokowymi operacjami logicznymi, arytmetycznymi, bitowymi i przesunięć. Istniejące jawnie w 8086 nieliczne ich odpowiedniki są jednak bardziej efektywne – nie wprowadzają narzutów czasowych, związanych z wielokrotnym odczytem kodu rozkazów (wada ta usunięta została w procesorze M68010). Bardzo przydatnym rozkazem M68000 jest rozkaz CHK, ułatwiający kontrolę zakresu zmienności wskaźników elementów tablic. Nie ma on swojego odpowiednika w procesorze 8086.

Zamiast podsumowania

Wyciągnięcie wniosków z powyższego porównania pozostawiam czytelnikowi. Jeżeli chodzi o mnie, zdecydowanie wolę Motorolę, także z powodu nie omawianych w niniejszym artykule własności sprzętowych M68k. Nie jest jednak moją intencją nawoływanie do rezygnacji ze stosowania procesora 8086. Popularność w świecie, zrealizowane już w kraju opracowania, rozpowszechnienie komputera osobistego IBM PC opartego na bliższym 8088, kolosalna liczba istniejącego i dostępnego oprogramowania oraz przewidywana dostępność mniej czy bardziej rodzimych układów procesora przemawiają w jego obronie. Oby był bardziej obecny w naszym życiu.

Nie ograniczajmy się jednak wyłącznie do 8086 i jemu pochodnych!!! Kiedyś możemy tego gorzko żałować.

---

1) W środowisku przyjęło się skrótowe określenie rodziny M68000 mianem M68k.

ADAM FORYCKI

Porównanie możliwych kombinacji trybów adresowania procesora M68000 – M i odpowiadających im trybów procesora 8086 – (I)

Tryb adresowania argumentu ↓ źródłowego	→docelowego	1.	2.	3.	4.	5.	6.	7.	8.	9.	
		8 (4)	8 (4)	8 (4)	8 (–)	8 (–)	8 (4)*	128 (4)*	1 (1)	1 (–)	Liczba opcji
1. Rejestrowy	Dn	MI	MI	MI	M	M	MI	MI	MI	M	8 (4)
2. Rejestrowy	An	MI	MI	MI	M	M	MI	MI	MI	M	8 (4)
3. Pośredni rejestrowy	(An)	MI	MI	MI	M	M	M	M	M	M	8 (4)
4. Pośredni rejestrowy z postinkrementacją	(An)+	M	M	M	M	M	M	M	M	M	8 (–)
5. Pośredni rejestrowy z predekrementacją	–(An)	M	M	M	M	M	M	M	M	M	8 (–)
6. Pośredni rejestrowy z wyrównaniem	d16(An)	MI	MI	M	M	M	M	M	M	M	8 (4)*
7. Pośredni indeksowy z wyrównaniem	d8(An,Xn)	MI	MI	M	M	M	M	M	M	M	128 (4)*
8. Bezwzględnie krótki	Abs.W	MI	MI	M	M	M	M	M	M	M	1 (1)
9. Bezwzględnie długi	Abs.L	M	M	M	M	M	M	M	M	M	1 (–)
10. Względny	d16(PC)	M	M	M	M	M	M	M	M	M	1 (–)
11. Względny indeksowy	d8(PC,Xn)	M	M	M	M	M	M	M	M	M	16 (–)
12. Bezpośredni	xxxx	MI	MI	MI	M	M	MI	MI	MI	M	1 (1)

* – w 8086 istnieją dwie wersje trybu – z krótkim (d8) i długim (d16) wyrównaniem

CZY PRZEWAGA TECHNOLOGICZNA WYSTARCZY?

Firma ATARI, znana dotąd jako producent komputerów domowych, postanowiła stworzyć nową jakość na rynku komputerów profesjonalnych. Pod względem technologicznym ATARI 520 ST ma przewagę nad dowolnym mikrokomputerem klasy PC. Ale czy sama technologia wystarczy do osiągnięcia sukcesu?

Na przełomie lat 1984–1985 Jack Tramiel. były szef firmy Commodore i „ojciec" C-64, oznajmił, że zamierza wprowadzić na rynek szybki mikrokomputer oparty na mikroprocesorze Motorola 68000. W końcu 1985 roku można już było pisać o tym jako o fakcie dokonanym. Za około 1300 dolarów amerykańskich można było zakupić kompletne ATARI 520 ST z pamięcią operacyjną 512 KB, monitorem monochromatycznym, jednostką dyskietkową oraz myszką.

WRAŻENIE OGÓLNE

Zanim włączymy 520 ST musimy się solidnie napracować. W przeciwieństwie do mikrokomputera Macintosh firmy Apple, w „Jackintoshu" każde urządzenie jest w oddzielnej obudowie i ma własny zasilacz. Spowoduje to niewątpliwie powstanie plątaniny kabli. Z pewnością wkrótce pojawi się na rynku firma, oferująca podstawkę do monitora, w której będzie miejsce na kable łączące poszczególne urządzenia. Mimo to montaż przebiega gładko, ponieważ mylne połączenie urządzeń nie jest możliwe.Wszystkie wejścia są opisane i opatrzone stosownymi symbolami.

Elegancka obudowa ma kolor beżowy, a klawiatura jest dobrej jakości i wygodna w obsłudze. Oprócz standarowego bloku alfanumerycznego, umieszczono trzy dodatkowe bloki klawiszy. Z prawej strony znajduje się blok numeryczny, a obok niego blok sterowania kursorem, zawierający również klawisze: Insert, Delete, Backspace i Help. Powyżej bloku alfanumerycznego umieszczono blok dziesięciu klawiszy funkcyjnych. Całość wygląda solidnie i funkcjonalnie, podkreślając tym samym ambicje producenta, który chce wejść przebojem na bardzo już wymagający rynek.

JEDNOSTKA CENTRALNA

Po otwarciu obudowy widać gęsto upakowaną, ale starannie wykonaną i dobrze rozplanowaną płytę główną. Sercem lub raczej mózgiem komputera jest mikroprocesor Motorola 68000. Dotychczas mikroprocesor ten zastosowano tylko w mikrokomputerach Macintosh Apple'a, Amiga Commodore, QL Sinclaira oraz w tzw. supermicro, np. SAGE. Procesor ten ma strukturę 32-bitową i 16-bitową szynę danych. Może on adresować bezpośrednio 16 MB pamięci, ale zastosowany układ MMU przewiduje maksymalnie 4 MB. Parametry te są zdecydowanie lepsze niż np. w mikroprocesorze Intel 8088, stosowanym w mikrokomputerach klasy PC.

Na płycie głównej 520 ST znajduje się 16 kostek pamięci. Są to układy o pojemności 256 Kb każdy, które razem tworzą 512 KB pamięci RAM.

Trochę gorzej rozwiązano pamięć ROM. W miejscu przewidywanym na 192 KB ROM na przełomie lat 1985–1986 znajdowała się tylko jedna kostka, zawierająca procedurę załadowczą wczytywanego z dyskietki systemu operacyjnego. Dziś komputery 520 ST są dostępne z pamięcią ROM (192 KB), zawierającą już prawie bezbłędną wersję TOS-u. Obok sterownika dyskietki na płycie głównej znajduje się układ DMA, który umożliwia bezpośrednie przyłączenie dysku Winchester. Niestety Winchester musi mieć własny układ kontrolera dysku. Przepustowość tego interfejsu dzięki zastosowaniu dodatkowej kostki DMA (Direct Memory Access) wynosi 1,3 MB na sekundę. 520 ST jest wyposażony również w układ akustyczny, emitujący trzy niezależne tony w obszarze przekraczającym zakres ludzkiego słuchu. Interfejs MIDI (Musical Instrument Digital Interface – interfejs cyfrowy dla instrumentów muzycznych) umożliwia przyłączenie syntezatora oraz innych instrumentów.

MONITOR EKRANOWY I JEDNOSTKA DYSKIETEK

Zgodnie z zapowiedziami Atari 520 ST został wyposażony w przyłącze do telewizora. Rozdzielczość monochromatycznego monitora (jedna z 3 możliwych) wynosi jednak 640x400 punktów, co czyni przyłączenie telewizora lub monitora typu Neptun 156 niemożliwym. Cena monitora kolorowego do 520 ST wynosi ok. 350 dolarów. Na monitorze tym można uzyskać do 512 kolorów. Przy czterech kolorach rozdzielczość wynosi 640x200 punktów, natomiast przy szesnastu – 320x200 punktów. Rozdzielczości te (zwane średnią i niską) mogą być stosowane na zwykłym telewizorze.

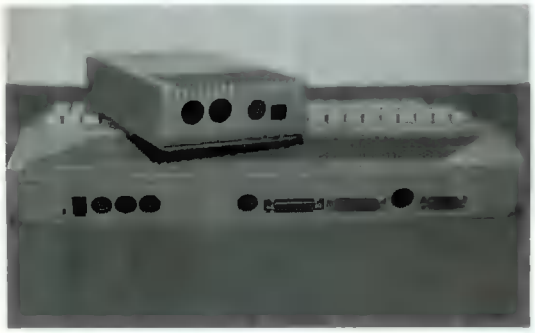
3,5-calowa jednostka dyskietek jest wprawdzie poręczna, ale ma pojemność tylko

357 kB w wersji jednostronnej i 726 kB w wersji dwustronnej, co nie zawsze wystarcza dla komputera wyposażonego w pamięć operacyjną o pojemności 512 KB. W obu wersjach jest zastosowany system zapisu o podwójnej gęstości (FM), na 80 ścieżkach.

OPROGRAMOWANIE

Wraz ze sprzętem są dostarczane języki Logo oraz BASIC i programy użytkowe 1-st Word i NEOChrom.

Po włączeniu monitora, dyskietki i komputera, procedura załadowcza ściąga system operacyjny z dyskietki, co trwa około pół minuty. Dla wersji komputera zawierającej system w pamięci ROM czas ten oczywiście jest znacznie krótszy (około 5 s). Następnie zgłasza się program GEM (Graphics Environment Manager – zarządzanie środowiskiem grafiki), opracowany przez firmę Digital Research jako system operacyjny wzorowany na oprogramowaniu Macintosha. GEM opiera się na połączeniu czterech elementów: techniki okienek, przewijanych menu, ikon oraz myszki. Specjalny programik szkoleniowy ułatwia zapoznanie się z obsługą myszki i możliwościami oferowanymi przez GEM w zakresie techniki okienek. Okienka mogą być na ekranie dowolnie zmniejszane, powiększane lub przemieszczane. Dane prezentowane w okienku mogą być przewijane w dowolnym kierunku. Przy wszystkich tych operacjach zaskakuje szybkość, z jaką są one wykonywane – uwidacznia się tu wyraźnie szybkość procesora. Korzystne jest również to, że z menu można bez obawy wybierać funkcje, ponieważ zawsze istnieje możliwość ich anulowania.



Model 520 ST jest dobrze wyposażony w interfejsy, co zapewnia łatwe dołączenie urządzeń zewnętrznych

Również Logo wykorzystuje technikę okienek. Nie jest to wprawdzie język tak powszechnie stosowany jak BASIC, ale stale zyskuje na popularności. Logo służy nie tylko do rysowania obrazków, lecz zawiera również mechanizm deklarowania procedur, a tym samym umożliwia programowanie strukturalne. W Logo na Atari można pisać nawet złożone programy. Stosunkowo duże zapotrzebowanie pamięci przez Logo jest prawdopodobnie przyczyną tego, że nie udało się od razu umieścić w ROM systemu operacyjnego TOS (Tramiel Operating System) oraz GEM, BASIC i Logo.

Atari 520 ST, nawet w jego fazie rozwojowej z przełomu lat 1985–1986, jest sprzętem fascynującym, a w dodatku oferowanym po bardzo korzystnej cenie. Mikrokomputer ten będzie jeszcze bardziej atrakcyjny po wprowadzeniu na rynek dysku stałego 15 MB w cenie ok. 600 dolarów. W niedalekiej przyszłości należy również oczekiwać taniej drukarki oraz układu do odczytu (tzw. CD-ROM), czyli pamięci optycznych opartych na odtwarzaczu laserowym.

Oferta Atari 520 ST w zakresie oprogramowania z każdą chwilą rośnie. Dziś istnieje już ponad 600 programów na Atari. Firma Digital Research wykonała dla Atari adaptację pakietów GEM-Write i GEM-Draw, natomiast sam producent sprzętu przystosował do ST stary, ale stosunkowo wydajny pakiet Atariwriter. Firma MicroPro ma wykonać odpowiednią wersję procesora tekstów WordStar. Również mniejsi producenci oprogramowania prowadzą prace w tym zakresie. Mimo takiej sytuacji firma Atari optymistycznie ocenia swoje perspektywy rynkowe.

Ale nawet gdyby znacznie więcej producentów oprogramowania zaangażowało się w produkcję pakietów dla Atari 520 ST, to w dalszym ciągu oferta ta będzie mizerna w porównaniu do biblioteki PC/MS-DOS. Ciekawym rozwiązaniem tego problemu jest awizowane wprowadzenie do sprzedaży układu, umożliwiającego eksploatację oprogramowania mikrokomputerów typu PC na mikrokomputerze Atari 520 ST, przy niepogorszonych parametrach eksploatacyjnych.

Pod względem technologicznym i cenowym Atari 520 ST wyprzedza wyraźnie mikrokomputery klasy PC, jednakże czynniki te dla większości potencjalnych nabywców nie są decydujące. Zaoszczędzenie 500 dolarów w wypadku użytkowników profesjonalnych odgrywa mniejszą rolę niż liczba i jakość oprogramowania, co spowoduje, że wielu zainteresowanych powstrzyma się z zakupem lub zdecyduje się na zakup jednej z wersji mikrokomputera kompatybilnego z PC.

Z drugiej strony producenci oprogramowania zaangażują się poważnie w oprogramowanie ST dopiero wtedy, gdy liczba sprzedanych egzemplarzy zapewni im opłacalność sprzedaży własnych produktów. Jest to podstawowy dylemat, przed którym obecnie stoi firma Atari.

micro

UKŁADY BIOLOGICZNE

Naukowcy z uniwersytetu Carnegie-Mellon w Pittsburgu (USA) pracują obecnie nad dwoma układami elektroniki cyfrowej, których podstawą są substancje biologiczne. Są to:

- układ pamięci o gęstości upakowania 1 GB (10^9 bajtów) na 1 cm^2,
- bramka NAND, tysiąc razy mniejsza od obecnie stosowanych.

Układ pamięci oparto na białku o nazwie bacteriorhodopsin, wytwarzanym przez genetycznie zmodyfikowany gatunek bakterii znalezionych w zatoce San Francisco. Białko to może występować w dwóch stanach, z których każdy pochłania światło z inną częstotliwością. Stan każdej cząsteczki białka może być odczytywany i zmieniany za pomocą lasera dostrojonego do jednej z tych częstotliwości.

W zbudowanym już układzie prototypowym trzy cząsteczki białka służą do przechowywania jednego bitu informacji. Taka trójka cząsteczek zajmuje powierzchnię ok. 100 angstremów kwadratowych*). Aczkolwiek gęstość układu prototypowego wynosi „zaledwie" 1 GB/cm^2, to teoretycznie możliwe są gęstości rzędu 100 000 GB/cm^2. Szokujące są tu jednak nie tylko wymiary układu, ale również szybkość przełączenia, wynosząca zaledwie 10 pikosekund na jedną operację. Należy zauważyć, że jest to o wiele mniej niż w dostępnych dziś modulatorach światła laserowego, od których zależy tu szybkość dostępu.

Na temat praktycznego zastosowania powyższego wynalazku naukowcy prowadzą rozmowy z firmą Seagate Technology.

Bramka NAND, stanowiąca drugi temat prac badawczych, będzie wymagała na wejściu i wyjściu również współpracy z laserem. Nie udało się jeszcze skonstruować prototypu, ale zespół badawczy przewiduje, że czas propagacji będzie wynosił ok. 3 pikosekund, a wymiary układu – 10 nanometrów na jedną bramkę. Prace badawcze nad tym układem są finansowane przez IBM.

*) Angstrem (A) – jednostka długości. 1 Å = 10^{-10}m = 10^{-1}nm

DRUKOWANIE BARWNE

Przeniesienie grafiki barwnej z ekranu na papier jest marzeniem wielu użytkowników sprzętu mikrokomputerowego. Przy cenie do ok. 400 dolarów, drukarki z możliwością wydruku barwnego stały się bardziej dostępne. W artykule omówimy różne techniki barwnego drukowania, a także dokonamy przeglądu tej kategorii drukarek, które nie przekraczają ceny 1500 dolarów. Chociaż w obecnych warunkach w Polsce sprzęt ten może mieć jeszcze ograniczony krąg użytkowników, sądzimy, że zapoznanie się ze stanem rozwoju technologicznego drukarek do grafiki barwnej i z aktualną ofertą rynkową będzie dla wielu Czytelników interesujące.

Większość pakietów oprogramowania graficznego, a także wiele języków programowania uwzględnia już wyprowadzanie grafiki barwnej na drukarce. Przy obecnych, coraz niższych cenach, można więc myśleć o zakupie drukarki przystosowanej do takiej grafiki. Na przykład drukarkę mozaikową typu GP-700A firmy Seikosha można kupić już za ok. 450 dolarów. Drukarka ta o szybkości 50 zn./s jest wprawdzie znacznie wolniejsza od niektórych modeli w wyższej kategorii cenowej (750-1500 dolarów), ale przy podejmowaniu decyzji zakupu szybkość ma tu znacznie mniejszy ciężar gatunkowy. Drukarka termiczna TPX-80 firmy C.Itoh działa z szybkością 80 zn./s, a więc nieco szybciej od poprzednio wymienionej, ale przy cenie 600 dolarów nie jest droższa w sposób odczuwalny dla potencjalnego nabywcy. Drukarkę Okimate 20 można kupić już za około 450 dolarów.

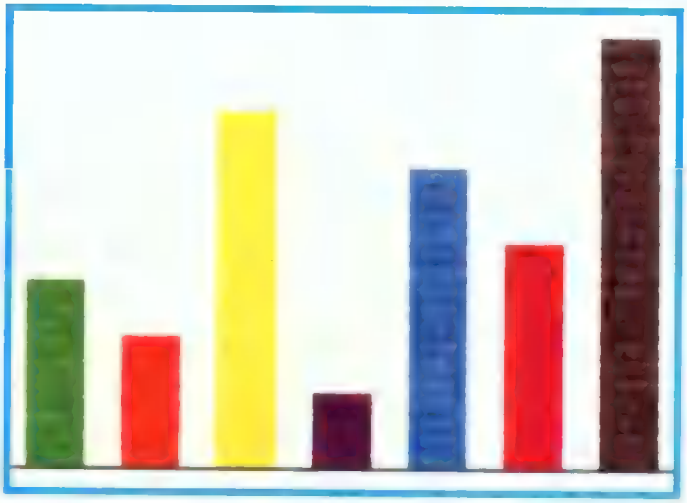
Trzy techniki drukowania

Bardzo istotną sprawą jest jakość wydruku barwnego. Zależy ona od sposobu przenoszenia farby na papier. Większość drukarek zawiera taśmę z wieloma pasmami barwnymi. Rozwiązanie to jest wprawdzie bardziej kosztowne, ale zapewnia lepszą jakość wydruku.

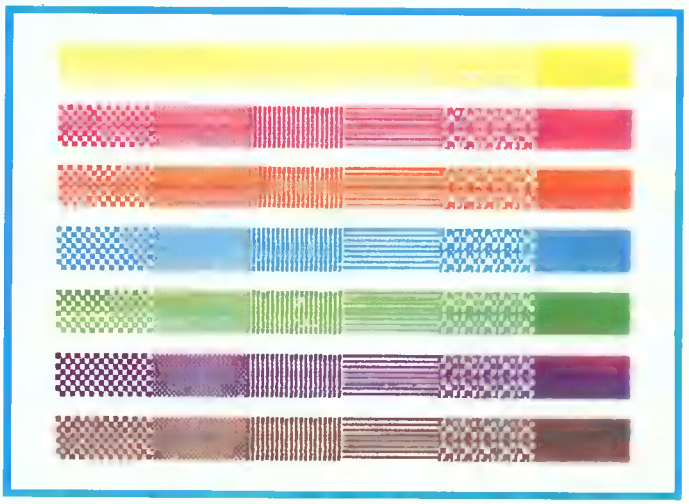
Do drukowania barwnego stosowane są następujące rodzaje drukarek:

- mozaikowe,
- termiczne,
- strumieniowe.

Wszystkie metody drukowania barwnego wykorzystują prawa kolorystyki, zgodnie z którymi z mieszaniny barw podstawowych tworzy się barwy pochodne. Z trzech barw podstawowych: czerwonej, żółtej i niebieskiej można przez odpowiednie mieszanie otrzymać w zasadzie wszystkie inne barwy. Decyduje tu procentowy udział każdej z barw podstawowych. Oczywiście mieszanie za pomocą drukarki nie jest tak proste jak na palecie malarskiej. Mieszanie barw w drukarkach polega najczęściej na tym, że barwy podstawowe są drukowane jedna na drugiej. Powoduje to, że liczba barw pochodnych jest bardzo ograniczona, ponieważ udział procentowy każdej z barw podstawowych jest zawsze taki sam. Stąd skala możliwości obejmuje tylko następujące siedem barw: purpurowa, czerwona, żółta, zielona, lazurowa, niebieska i czarna. Niemożliwe jest np. zmieszanie małej ilości błękitu z dużą ilością czerwieni. Proces mieszania barw podsta-



Epson JX-80: Smugi na płaszczyznach barwnych są charakterystyczne dla wszystkich drukarek mozaikowych, bez względu na ich cenę



C. Itoh C 310: Drukarka mozaikowa z czterobarwną taśmą pozwala uzyskać następujące kolory: żółty, czerwony, pomarańczowy, niebieski, zielony, purpurowy i czarny

wowych jest realizowany w drukarkach w rozmaity sposób. Drukarki mozaikowe odwzorowują za pomocą igieł poszczególne punkty, które skupione w formie czworoboku tworzą jeden znak (pozycję drukarską). Siatka różnobarwnych punktów powoduje uzyskanie efektu nowej barwy (odcienia). Wspomniana już drukarka Okimate 20 umożliwia uzyskanie 18 różnych barw.

Drukarki mozaikowe

W drukarkach mozaikowych nośnikiem przenoszącym kolor są taśmy z wieloma posmami barw. Najczęściej pasma te mają trzy barwy podstawowe oraz czerń. Wskutek podnoszenia i opuszczania taśmy następuje ustawienie pasma odpowiedniej barwy przed igłami głowicy drukującej. W czasie drukowania jednego wiersza barwa pasma nie może być zmieniona. Dlatego wiersz, w którym mają być użyte wszystkie cztery barwy podstawowe, musi być czterokrotnie powtórzony, tak aby każdy z punktów uzyskał właściwą barwę. Metoda ta powoduje znaczne zmniejszenie efektywnej szybkości drukowania. Druk mozaikowy z użyciem taśmy wielobarwnej ma tę wadę, że pasma częściej używanych barw ulegają szybszemu zużyciu niż pozostałe. Spowoduje to oczywiście zmianę stosunku procentowego mieszania barw, a więc zniekształcenie zamierzonego efektu barwnego.

Większość drukarek do grafiki barwnej ma możliwość stosowania taśmy czarnej, co pozwala wykorzystywać je również do wydruków jednobarwnych.

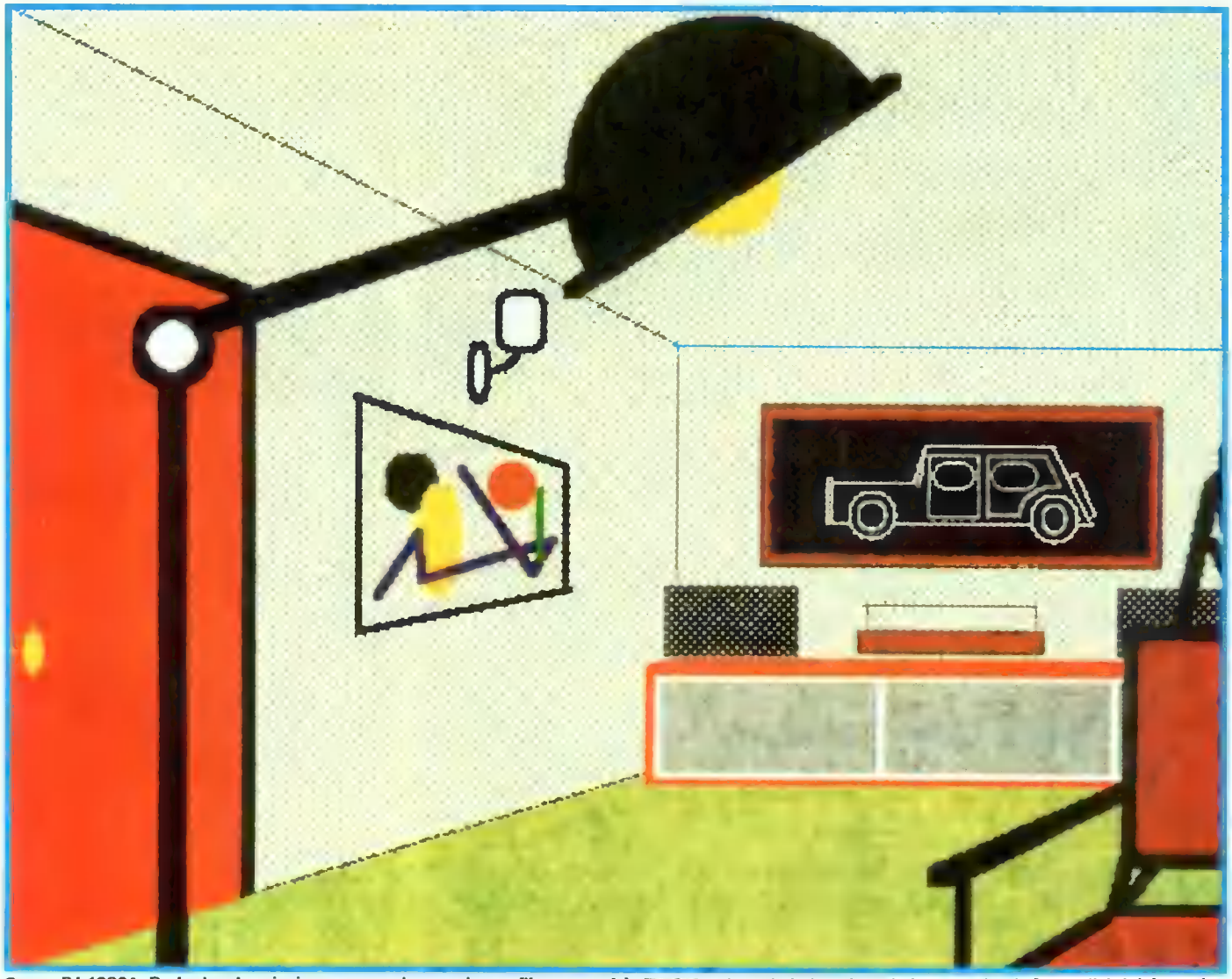
Taśmy barwne zużywają się zwykle po wykonaniu w jednym paśmie 1 mln znaków. W niektórych drukarkach (np. GP-700A) istnieje możliwość regenerowania taśm.

Wszystkie drukarki mozaikowe – zarówno tanie, jak i drogie – mają istotną wadę, że przy zmianie wierszy powstają smugi, szczególnie widoczne w wypadku drukowania płaszczyzn o tej samej barwie.

Drukarki termiczne

Drukarki termiczne działają na identycznej zasadzie jak mozaikowe (druk matrycą igłową). Pracują one prawie bezgłośnie, ponieważ zastosowano w nich technikę termicznego przenoszenia barwnika. Technika ta polega na tym, że podgrzane igłowe termoelementy głowicy drukującej powodują topienie wierzchniej warstwy woskowej taśmy barwiącej. Warstwa ta przylepia się zarówno do zwykłego papieru jak i przezroczystej folii, co oczywiście rozszerza możliwości zastosowań drukarki. Błyszczący wydruk przypomina efekty płaszczyzn lakierowanych. Podobnie jak w drukarkach mozaikowych, efekty barwne uzyskuje się przez podnoszenie i opuszczanie wielobarwnych taśm z równoległymi pasmami barw podstawowych. Stosowane są również taśmy, np. we wspomnianej już drukarce Okimate 20, gdzie pasma mają postać powtarzających się sekwencji pionowych barw podstawowych.

W drukarkach termicznych efekt barwny powstaje również przez kolejne nakładanie barw podstawowych i rastrowanie, uzyskane w wyniku punktowego działania głowicy drukującej za pomocą igieł. Podstawową wadą drukarek termicznych jest to, że miejsca taśmy, w których została stopiona warstwa wosku, nie mogą być ponownie wykorzystane. Taśma zużywa się tu więc znacznie szybciej niż w drukarce mozaikowej,



Canon PJ-1080A: Drukarka strumieniowa przenosi na papier grafikę wprawdzie bardzo wolno, ale kolorystycznie bardzo wiernie i co najistotniejsze – bez jakichkolwiek smug

Symbol wyrobu	Producent	Metoda barwienia	Elementy drukujące	Liczba barw: wyjściowych	Liczba barw: wydruku	Żywotność nośnika barw	Szybkość drukowania (zn./s)	Orientacyjna cena (dolary)
1550 SCP/SCEP/	C.Itoh	taśma	igieł	3	7	1 mln znaków/pasmo	180	1400
1550 SCR	C.Itoh	taśma	9 igieł	3	7	1 mln znaków/pasmo	180	1500
8510 SCP/SCEP	C.Itoh	taśma	igieł	3	7	1 mln znaków/pasmo	180	1150
C-310 CP/CEP	C.Itoh	taśma	igieł	4	7	2 mln znaków/pasmo	250	1250
C-310 CR	C.Itoh	taśma	igieł	4	7	2 mln znaków/pasmo	250	1250
GP-700A	Seikosha	taśma	8 igieł	4	7	2 mln znaków/pasmo	50	450
Imagewriter	Apple	taśma	–	4	7	–	180	1110
Juki 5520	Juki	taśma	igieł	4	dowolna	1 mln znaków/pasmo	180	900-1050
JX-80	Seiko Epson	taśma	igieł	4	7	0,8-1,4 mln znaków/pasmo	160	1000
Microline 293	OKI	taśma	18 igieł	4	14	3 mln znaków/pasmo	200	1200
Microline 292	OKI	taśma	18 igieł	4	14	3 mln znaków/pasmo	200	950
Pinwriter	NEC	taśma	2x12 igieł	4	7	mln znaków/pasmo	180	1320
Okimate 20	OKI	termiczna	24	3	16	stron A4	3 wiersze/min	440
TPX-80	C.Itoh	termiczna	24	3	7	400 wierszy		570
CGP-220	Tandy	natryskowa	4 dysze	4	7	3-4 mln znaków/zasobnik	37	1000
PJ-1080A	Canon	natryskowa	4 dysze	4	7	3,2 mln znaków/zasobnik		1130

np. w drukarce Okimate 20 wystarcza tylko do zapisania ośmiu stron formatu A4, koszt jej zakupu wynosi obecnie aż ok. dolarów.

Mimo tej wady, drukarki termiczne mają wiele zalet. Są szybsze, pracują bardzo cicho i są najtańsze. Uzyskane grafiki mają bardzo efektowny wygląd, że względu na intensywny połysk farb woskowych, Podobnie jak drukarkach mozaikowych, istotną wadą są wspomniane smugi na wydruku, mające swe źródło w sposobie tworzenia obrazu przez wielokrotne nakładanie barw taśmy drogą powtarzania przebiegów tego samego wiersza.

Drukarki strumieniowe

Drukarki strumieniowe działają na zasadzie natrysku farby drukarskiej papier. Technologia ta, stosowana już przy zapisie jednobarwnym, polega na wyposażeniu głowicy drukującej w dysze połączone zasobnikami farby. W drukarce do grafiki barwnej znajdują się cztery dysze z odrębnymi zasobnikami czterech farb.

Drukarki strumieniowe zapewniają bardzo wysoką jakość grafiki, wyrażającą się intensywnością barw oraz eliminacją smug wynikających stosowania taśmy. Poprawę jakości obrazu można uzyskać przez zastosowanie specjalnego papieru zwiększonej chłonności. Dla drukarki PS-1090A firmy Canon cztery 50-metrowe rolki takiego papieru kosztują ok. 45 dolarów.

Niezwykle precyzyjny mechanizm drukarki strumieniowej zapewnia wprawdzie bardzo wysoką jakość grafiki, ale wywiera również wpływ poziom cen, które kształtują się powyżej 1000 dolarów.

Podejmując decyzję zakupu drukarki do grafiki barwnej należy uwzględniać nie tylko jakość wydruku, szybkość drukowania oraz cenę, ale również upewnić się, czy drukarka i komputer mają taki sam interfejs.

Nie wyczerpuje to oczywiście problemu, ponieważ do wykorzystania drukarki jest potrzebne odpowiednie jej oprogramowanie. Niektórzy producenci oferują takie oprogramowanie dostosowane do najpopularniejszych modeli mikrokomputerów. Również wiele pakietów graficznych zawiera niezbędne oprogramowanie drukarki. Użytkownik dysponujący pakietem graficznym oprogramowanie takie może wykonać sam. Konieczna jest wówczas dobra znajomość zasad działania konkretnej drukarki i niezbędnych parametrów instrukcji jej obsługi.

Załączony przegląd drukarek oparto na wynikach ankiety rozesłanej do najbardziej znanych producentów tej kategorii sprzętu i zapewne nie obejmuje wszystkich modeli, jakie wciąż pojawiają się na rynku. Ujęto w nim jedynie drukarki w cenie poniżej 1500 dolarów. W zestawieniu podano najważniejsze parametry techniczne i użytkowe poszczególnych modeli oraz ich orientacyjne ceny.

PUSTE BIURKO

Prawie żaden użytkownik mikrokomputera nie może obejść się bez notesu, kalkulatora lub kalendarza. Wszystkie te urządzenia zastępuje stosunkowo tani pakiet programów – SideKick, przeznaczony dla IBM PC i mikrokomputerów kompatybilnych.

Podstawowa zaleta SideKicka polega na tym, że jest on stale dostępny dla użytkownika. Po włączeniu mikrokomputera wystarczy wprowadzić z klawiatury symbol **SK**, co powoduje zainstalowanie SideKicka w pamięci operacyjnej. W ten sposób, aż do wyłączenia mikrokomputera mamy do dyspozycji m.in. edytor tekstów, kalkulator oraz kalendarz. SideKick, zależnie od wygenerowanej wersji, zajmuje 30-40 KB pamięci operacyjnej.

Zarówno po zainstalowaniu pakietu, jak też przy każdym jego wywołaniu na ekranie pojawia się menu, umożliwiające wybranie dowolnego modułu SideKicka. Aby wywołać pakiet podczas działania innego programu, należy nacisnąć jednocześnie klawisze CTRL i ALT, co spowoduje przerwanie wykonywanego programu i pojawienie się na ekranie okienka z głównym menu SideKicka. Niestety SideKick nie może być wywoływany podczas działania programów, które same obsługują przerwania klawiatury. Główne menu pakietu zawiera następujące moduły wybrane naciśnięciem odpowiedniego klawisza funkcyjnego (F1-F7) lub litery będącej skrótem nazwy:

- F1 – **Instrukcja** (H – Help),
- F2 – **Notes** (N – NotePad),
- F3 – **Kalkulator** (C – Calculator),
- F4 – **Kalendarz** (L – caLendar),
- F5 – **Centralka telefoniczna** (D – Dialer),
- F6 – **Tabela ASCII** (A – ASCII – tabele),
- F7 – **Montaż** (S – Setup).

Po zakończeniu korzystania z SideKicka można kontynuować działanie przerwanego programu.

Poszczególne moduły SideKicka działają w niezależnych okienkach, które mogą na siebie zachodzić oraz być powiększane, zmniejszane lub dowolnie przesuwane (po naciśnięciu klawisza SCROLL-LOCK), m.in. w celu obejrzenia zachowanej w tle zawartości przerwanego programu. Moduł **Instrukcja** (Help) wyświetla odpowiednie fragmenty instrukcji obsługi pakietu. Podczas pracy z dowolnym z modułów SideKicka można powrócić do głównego menu naciskając klawisz ALT lub przejść do innego modułu naciskając jednocześnie klawisze ALT i litery symbolu modułu.

Edytor tekstów

Najważniejszym modułem SideKicka jest **Notes** (Notepad). Jest to kompletny edytor, zawierający również szereg funkcji procesora tekstów. Instrukcje redakcyjne są takie same, jak w pakiecie WordStar, co sprawia, że większość użytkowników nie będzie miała trudności z opanowaniem obsługi **Notesu.** Bardziej wybredni mogą za pomocą dostarczonego z pakietem programu zmodyfikować poszczególne instrukcje, stosownie do swoich przyzwyczajeń.

Edytor umożliwia zapamiętanie wprowadzonego tekstu lub zapisanie go do istniejącego zbioru. Oczywiście możliwe jest również wczytywanie tekstów z dyskietki. Jest to szczególnie wygodne wtedy, gdy podczas pracy w procesorze tekstów chcemy obejrzeć lub poprawić inny tekst.

Jeżeli na początku redagowanego tekstu znajduje się instrukcja LOG, przy każdym wywołaniu edytora na końcu znajdującego się w pamięci tekstu zostaje automatycznie zapisana aktualna data i godzina. Dzięki temu jest możliwe późniejsze wydrukowanie zapisanych notatek. Ten sam efekt można uzyskać przez naciśnięcie klawiszy CTRL-Q-T.

Jedną z najważniejszych zalet edytora jest możliwość wpisania (do obszaru pamięci przeznaczonego na redagowanie) całości lub części zawartości ekranu, wyświetlanej w momencie wywołania SideKicka. Opcja ta jest przydatna do dokumentowania programów lub zapamiętywania wybranych danych. Wielkość okienka edytora może być kształtowana dowolnie. Ponieważ edytor zapewnia również poziome przewijanie ekranu – długość wiersza nie jest ograniczona wielkością okienka. Jedynym mankamentem edytora jest brak automatycznego formatowania i przenoszenia wyrazów. Opcja ta została świadomie pominięta przez producenta po to, aby uniknąć w tekście specjalnych znaków sterujących.

Kalkulator

Podczas pracy na mikrokomputerze często konieczne jest wykonanie obliczeń. Zapewnia to moduł **Kalkulator**. Obsługa tego modułu jest zrozumiała dla każdego, kto potrafi posługiwać się kieszonkowym kalkulatorem. **Kalkulator** SideKicka umożliwia przechowywanie danych w pamięci buforowej, a także wykonywanie złożonych operacji arytmetycznych z użyciem nawiasów.

Kalkulator umożliwia wykonywanie obliczeń w zakresie liczb zawierających dwanaście miejsc przed i cztery po przecinku oraz pracuje w systemie BCD, który wyklucza pojawienie się błędów zaokrągleń. Po wywołaniu **Kalkulator** przełącza automatycznie klawiaturę komputera na tryb NUM-LOCK, co powoduje, że pole kursora i cyfr akceptuje wyłącznie wprowadzanie danych numerycznych.

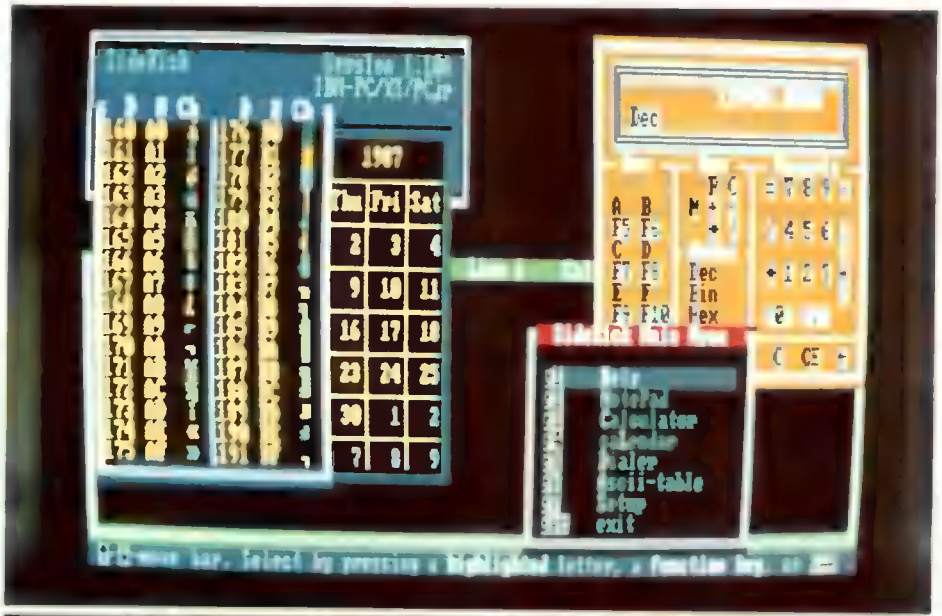
Kalkulator jest szczególnie użyteczny dla programistów, ponieważ umożliwia wykonywanie obliczeń w systemie szesnastkowym, dwójkowym lub dziesiętnym. Przechodzenie z jednego systemu na drugi odbywa się przez naciśnięcie klawisza, co eliminuje konieczność często potrzebnych przy programowaniu przeliczeń. **Kalkulator** jest ponadto wyposażony w funkcje logiczne AND, OR i XOR, umożliwiające symulowanie funkcji procesora.

Niestety nie jest możliwe ani zapamiętywanie wyników, ani też wprowadzanie ich do redagowanego tekstu. Nie jest również możliwe zapamiętywanie wzorów powtarzających się operacji, w których zmieniają się tylko pewne parametry. Możliwość taką zapewnia jak wiadomo wiele programowanych kalkulatorów kieszonkowych.

Kalendarz

Kalendarz jest uruchamiany podaniem daty. Umożliwia on zapisanie terminów z dokładnością do pół godziny, w przedziale od ósmej rano do ósmej wieczorem. Może on być również wykorzystany do ustalenia dnia tygodnia dla dowolnej daty. Za pomocą klawisza funkcyjnego F2 można wczytać do pamięci inny kalendarz. Umożliwia to np. sekretarce jednoczesne zapamiętywanie terminów spotkań dla wielu osób korzystających z tego samego mikrokomputera. Poszczególne części kalendarza (np. wszystkie dni z terminami spotkań) można wydrukować, a każdy z terminów skasować lub zmienić.

Moduł **Centralka Telefoniczna** współpracuje tylko z modemami kompatybilnymi z urządzeniami produkowanymi przez firmę Hayes.



Rys. 1. Sześć okienek SideKicka

Rys. 2. CTRL i ALT wywołują na ekran główne menu SideKicka

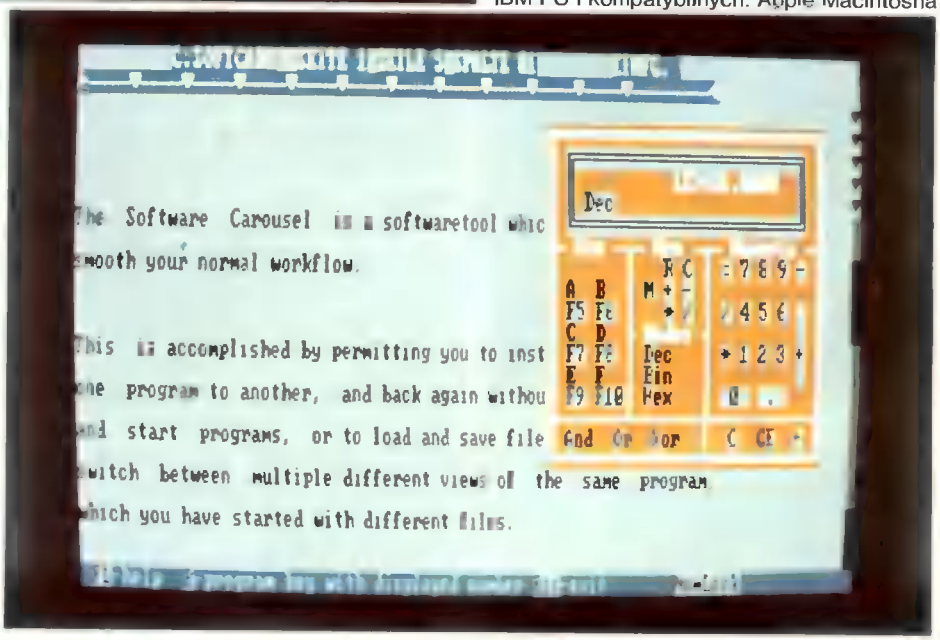
W RFN wskutek niedopuszczenia ich do eksploatacji, a także braku modemów z automatycznym wybieraniem numerów, moduł ten w niemieckiej wersji pakietu usunięto.

Centralka telefoniczna jest spisem telefonow użytkownika zawierającym trzypolowe rekordy. W każdym z nich zapisane są: kod, numer telefonu i komentarz (np. adres). Spis ten, zakładany i modyfikowany za pomocą edytora, jest umieszczony na dyskietce pod nazwą **PHONE.DIR**, która może być zmieniona w module **SETUP**. Po podaniu kodu następuje automatyczne wybranie odpowiadającego mu numeru telefonu. Można również podawać i wybierać numer telefonu bezpośrednio z redagowanego tekstu za pomocą kursora.

Większość programistów używa tabeli zawierającej zestaw znaków komputera oraz odpowiadające im wartości szesnastkowe i dziesiętne. Dlatego też SideKicka wyposażono w moduł **Tabela kodu ASCII**. Zawiera on oprócz wyżej wymienionych danych, również znaki oraz mnemonik każdego znaku odpowiadający poszczególnym wartościom CTRL.

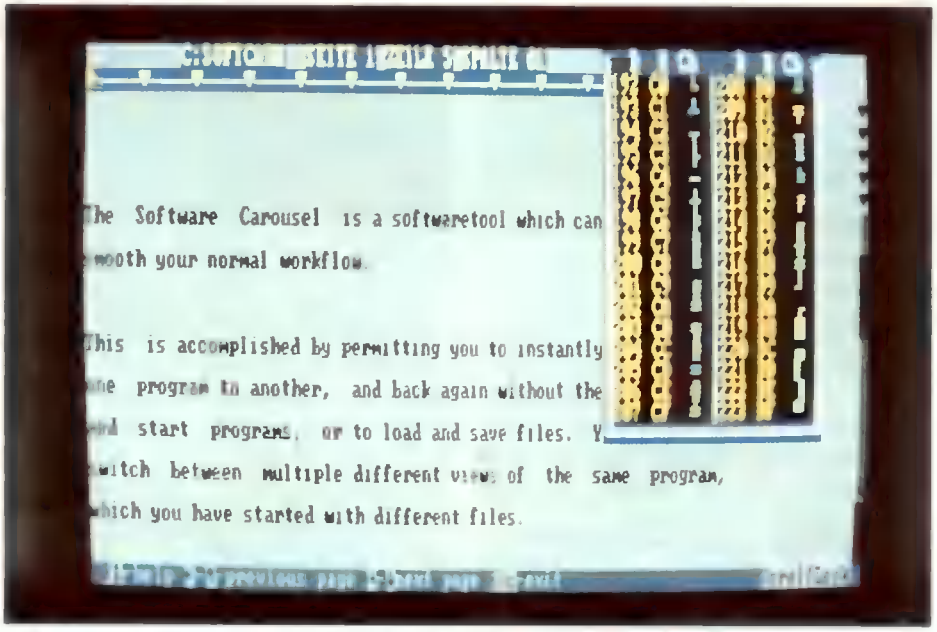
Moduł **Montaż** umożliwia między innymi zapamiętanie położenia poszczególnych okienek na ekranie. Dla bardziej złożonych instalacji istnieje odrębny program umożliwiający modyfikowanie instrukcji edytora lub zmiany maksymalnej wielkości **Notesu**.

SideKick jest pakietem znacznie usprawniającym pracę na mikrokomputerach typu IBM PC i kompatybilnych. Apple Macintosha



Rys. 3. Moduły SideKicka można w każdej chwili włączyć do aktualnie wykonywanego programu

Rys. 4. W KALENDARZU można zanotować termin z dokładnością do pół godziny



Rys. 5. Tabela ASCII umożliwia szybkie odnalezienie odpowiedniego znaku - w tle WordStar

chwalono między innymi za to, że miał wbudowany kalkulator i notes. Obecnie również każdy z użytkowników IBM PC ma możliwość korzystania z tych (i innych) opcji, które wprawdzie nie są tak efektowne jak w Apple, lecz za to racjonalniej zrealizowane. SideKick nie wymaga grafiki, można go więc stosować na dowolnym monitorze. Jeżeli dysponujemy monitorem kolorowym, to jego możliwości są automatycznie wykorzystywane (np. wielobarwne okienka). Dokumentacja, w postaci cienkiej książeczki kieszonkowego formatu, zawiera jasny i wyczerpujący opis wszystkich funkcji.

Prwdopodobnie niejeden użytkownik życzyłby sobie jednak wzbogacenia zakresu funkcji SideKicka. Ale nawet w aktualnej wersji jest to pakiet absolutnie niezbędny dla każdego użytkownika IBM PC – a tak powiedzieć można tylko o nielicznych pakietach, zwłaszcza tak tanich jak SideKick.

micro

ANKIETA

Zamieszczone w ubiegłorocznym zeszycie 3 hasło „Kształtujmy wspólnie profil MIKROKLANU" przyniosło już znaczną liczbę interesujących wypowiedzi najbardziej aktywnych Czytelników. Wypowiedzi te starannie analizujemy, a niektóre uwagi i pomysły uwzględniamy już w bieżącej pracy redakcyjnej. Jest ich jednak zbyt mało, aby były reprezentatywne dla opinii i życzeń odbiorców stutysięcznego nakładu.

Aby uzyskać w pełni wiarygodny obraz struktury całej społeczności Czytelników, a zwłaszcza jej aktualnych potrzeb informacyjnych, postanowiliśmy przeprowadzić ankietę. Jej wyniki pozwolą znacznie lepiej niż dotąd dostosować treść MIKROKLANU do rzeczywistego, a tak szybko zmieniającego się, zapotrzebowania odbiorców czasopisma mikrokomputerowego. Nie trzeba podkreślać jak jest to istotne dla jego przyszłego rozwoju. Wiadomo jednak, że o wiarygodności wyników każdej ankiety decyduje liczba nadesłanych odpowiedzi. Aby zachęcić Czytelników do udziału w ankiecie, redakcja przewiduje rozlosowanie wśród respondentów atrakcyjnych nagród rzeczowych.

Treść ankiety, szczegółową listę nagród oraz warunki uczestnictwa podamy w następnym numerze.

Warunkiem udziału w losowaniu jest wycięcie i zachowanie kuponu wydrukowanego na ostatniej stronie okładki.

CZYM JEST UNIX?

JĄDRO

Jądro jest najbardziej wewnętrzną częścią Unixa, która realizuje podstawowe funkcje sterowania i nadzoru wejścia-wyjścia, zarządzania plikami i procesami oraz gospodarki pamięcią. Jądro stanowi jedyną część systemu, do której użytkownik nie ma bezpośredniego dostępu i nie może wymienić jej na własną wersję.

W skład jądra Unixa wchodzą trzy główne części:
– część zasadnicza, napisana w języku C (p. „Informatyka", nr 4-6, 1985), obejmująca ok. 10 000 linii programu, stanowiąca tworzywo, z którego mogą być budowane polecenia,
– kod operacji elementarnych (ang. machine primitives), napisany w języku asemblera, obejmujący ok. 1000 instrukcji, realizujący funkcje zbliżone do poziomu sprzętowego, wymagające bardzo dużej efektywności,
– kod obsługi urządzeń (ang. device driver code).

Z punktu widzenia opisu użytkowego najważniejsza jest część podstawowa, stanowiąca treść tzw. wywołań systemowych, będących samodzielnymi procedurami o ściśle sprecyzowanych funkcjach i parametrach. Wywołania systemowe są dostępne oczywiście z języka asemblera, ale także z języka C (to jest ich najczęstszy sposób użycia) i z Fortranu 77. W poniższym opisie przedstawiamy tylko tę część jądra, rezygnując z bardziej wyspecjalizowanych programów operacji elementarnych i programów obsługi urządzeń.

Wejście-wyjście

Podsystem wejścia-wyjścia dzieli się na dwie części:
– obsługę urządzeń operujących blokami informacji (jak np. dyski),
– obsługę urządzeń operujących ciągami znaków (np. monitory, drukarki itp.).

W celu uzyskania maksymalnej jednorodności systemu zapewniono elementarne operacje wejścia-wyjścia, dotyczące m.in. tworzenia, otwierania, odczytu, zapisu, przeszukiwania i zamykania, identyczne dla wszystkich typów urządzeń.

Operacja otwierania pliku ma postać:

filep = open (name, flag)

gdzie **name** jest nazwą pliku, **flag** – parametrem wskazującym, czy plik ma być odczytany bądź zapisywany, a **filep** – wartością udostępnioną przez operację **open**, będącą małą liczbą naturalną, służącą jako indeks do jednoznacznej identyfikacji pliku. Operacja otwierania pliku ma zadanie przekształcenia jego nazwy na odpowiedni opis pliku. Przekształcenie to rozpoczyna się od opisu jednego ze skorowidzów, najczęściej skorowidza danego użytkownika. Jeżeli dany plik nie jest do niego przypisany, to przeszukuje się kolejne elementy hierarchicznej struktury plików. Po znalezieniu odpowiedniego opisu umieszcza się go w segmencie danych systemu w tablicy aktywnych opisów plików.

Do tworzenia pliku służy operacja **create**, której uproszczona postać jest następująca:

filep = create (name)

Parametry **name** i **filep** mają identyczne znaczenie jak w operacji **open**. W trakcie realizacji wywołania **create** system tworzy opis nowego pliku oraz w odpowiednim skorowidzu, do którego jest przypisany nowo tworzony plik, wstawia parę danych: nazwę pliku oraz indeks tablicy wskazujący opis tego pliku. Operacja **create** dodatkowo otwiera ten plik w trybie zapisu.

Do realizacji odczytu i zapisu służą operacje **read** i **write**. Mają one następującą postać:

n = read (filep, buffer, count)
n = write (filep, buffer, count)

gdzie **filep** jest indeksem pliku nadanym przez operację **open** lub **create**, **buffer** – buforem interpretowanym jako tablica znaków, **count** – liczbą bajtów, które mają być odczytane (lub zapisane), a **n** – wartością udostępnianą przez operacje, określającą liczbę bajtów faktycznie odczytanych lub zapisanych. Jeżeli operacja zakończyła się poprawnie to **n=count**. W szczególności, jeżeli przy odczycie wystąpiła sytuacja **n=0**, to oznacza osiągnięcie końca pliku. Odczyt i zapis traktuje się jako operacje sekwencyjne, o ile nie jest to explicite realizowane inaczej. Dla każdego pliku system przechowuje wskaźnik kolejnego bajtu do odczytu lub miejsca do zapisu.

Do zmiany wartości tego wskaźnika służy operacja **lseek** o postaci:

location = lseek (filep, offset, base)

gdzie **filep** jest indeksem pliku nadanym przez operację **open** lub **create, base** – bazą, w stosunku do której określa się nową wartość wskaźnika odczytu lub zapisu (baza może przyjmować jako wartość początek lub koniec pliku, lub aktualne położenie wskaźnika), **offset** – nowym położeniem wskaźnika (w bajtach) w odniesieniu do bazy, **location** – nowym położeniem wskaźnika po wykonaniu operacji **lseek** w odniesieniu do początku pliku. Operacja **lseek** umożliwia realizację bezpośredniego dostępu do danych na dyskach. Dla niektórych urządzeń, np. drukarek i monitorów, jest ona zabroniona.

Oprócz operacji wymienionych powyżej w Unixie występuje cały szereg innych. Realizują one takie funkcje jak: zamykanie, kasowanie, zmiana właściciela pliku, zmiana praw dostępu do pliku (odczytywanie, zapisywanie, wykonanie), przyłączenie się do pliku już istniejącego.

Procesy

Proces jest wykonaniem tzw. obrazu, przy czym obraz (ang. image) jest definiowany jako aktualny stan środkowiska obliczeniowego, na które składa się obraz pamięci, rejestrów, stan otwartych plików i bieżący skorowidz. Inaczej mówiąc obraz jest aktualnym stanem wirtualnego komputera.

W obrazie pamięci procesu wyróżnia się trzy segmenty: segment tekstu programu, segment danych i segment stosu.

Segment tekstu programu rozpoczyna się od adresu 0 w wirtualnej przestrzeni adresowej. W segmencie tym nie można niczego zapisywać, można go tylko odczytać. Jest on dzielony między różne procesy wykonujące ten sam kod.

W drugim segmencie znajdującym się za tekstem programu mieszczą się dane. W tym obszarze proces może je zarówno zapisywać jak i odczytywać. Znajdują się tu dane użytkownika oraz niewielki segment danych systemowych o stałych rozmiarach, który zawiera informacje o aktywnym procesie.

W trzecim fragmencie pamięci przechowuje się stos. Rozpoczyna się on od największego adresu w pamięci wirtualnej procesu. W miarę zapełniania stosu jego wierzchołek przesuwa się w kierunku mniejszych adresów.

W obszarze pamięci procesu nie ma buforów plików, które znajdują się w jądrze systemu.

Do organizacji przepływu sterowania między procesami służy zbiór operacji elementarnych, z których najważniejsze scharakteryzowano poniżej.

Do powoływania nowego procesu służy operacja rozwidlania (ang. fork):

processid = fork ()

gdzie **processid** jest jednoznacznym identyfikatorem procesu. Proces powołujący inny proces nazywa się procesem macierzystym (ang. parent process), a proces powoływany – procesem potomnym (ang. child process). Każdy proces ma swój proces macierzysty. Jedyny wyjątek od tej reguły występuje przy inicjalnym ładowaniu systemu (tworzeniu procesu powłoki). Proces potomny dziedziczy po procesie macierzystym kopię obrazu pamięci. Dla obu procesów dostępne są te same pliki i ewentualnie segment programu.

Inną elementarną operacją jest:

execute (file, arg_1, arg_2, ..., arg_n)

gdzie **file** jest plikiem zawierającym polecenia Unixa (skrypt), a **arg_1, arg_2, ..., arg_n** – wartościami parametrów tego polecenia, będącymi ciągami znaków. Operacja ta nie powoduje powołania nowego procesu, lecz wykonujący ją proces wymienia swoje segmenty tekstu programu oraz danych na segmenty odpowiadające poleceniom odczytanym z pliku **file.** Po wykonaniu operacji **execute** proces posiada nadal te same otwarte pliki, bieżący skorowidz oraz powiązania z innymi procesami.

Do komunikacji między procesami służy operacja

filep = pipe ()

gdzie **filep** jest indeksem utworzonego pliku. Operacja ta tworzy potok danych między procesami. Procesy odczytują (lub zapisują) dane, tak jak ze (lub do) zwykłego pliku. Proces odczytuje dane operacją **read,** tak jak z pliku o indeksie **filep,** a jeżeli w pliku tym nie ma dostatecznej liczby danych, to czeka, aż inny proces zapisze tu dostateczną liczbę bajtów za pomocą operacji **write**. Przy powoływaniu nowego procesu operacją **fork** potok jest dziedziczony przez proces potomny.

Proces może zawiesić swoje działanie za pomocą operacji:

processid = wait (status)

gdzie **status** jest parametrem określającym rodzaj zdarzenia, które spowoduje wznowienie działania procesu, a **processid** – identyfikatorem procesu, który spowodował to zdarzenie.

Proces może zakończyć swoje działanie operacją:

exit (status)

Powoduje ona zakończenie wykonywania procesu, zamknięcie plików i zniszczenie obrazu. Wartość parametru **status** przekazuje się do procesu macierzystego, jeżeli wykonał on operację **wait (status).**

Polecenia dotyczące procesów

Oprócz wywołań systemowych użytkownik ma do dyspozycji także kilka poleceń bezpośrednio operujących procesami.

Do najważniejszych należy polecenie **ps** (ang. process status) powodujące wyświetlenie stanu aktywnych procesów. Przykładowo wydanie polecenia:

$ps –a

oznaczające żądanie podania stanu wszystkich procesów (opcja **a,** ang. all) może spowodować wyświetlenie następującej informacji:

PID	TTY	TIME	CMD
2106	02	0:00	-ps
2110	11	0:15	-sh

Jak łatwo się domyślić poszczególne kolumny wydruku oznaczają: identyfikator procesu **(PID,** ang. process id), terminal, z którego rozpoczęto proces (**TTY,** ang. teletype) czas trwania procesu (**TIME**) i polecenie wydane w celu zainicjowania procesu (**CMD,** ang. command).

Polecenie **at** powoduje zainicjowanie w określonym czasie procesu odpowiadającego danemu plikowi poleceń (skryptowi). Na przykład, aby odtworzyć hejnał mariacki w południe można wydać następujące polecenie:

$at 1200 hejnał

gdzie plik o nazwie **hejnał** zawiera zbiór poleceń uruchamiających odpowiednie urządzenia odtwarzające, nagłaśniające itp.

Polecenie **nice** umożliwia zmniejszenie priorytetu procesu w celu uniknięcia konfliktu z wykonywaniem innych poleceń z tego samego terminala, np.:

$nice cp plik_stary plik_nowy &

spowoduje zmniejszenie o 10 (domyślnie) priorytetu procesu odpowiadającego poleceniu **cp** wykonywanemu w tle.

Polecenie **nohup** (ang. no hang up, nie zawieszaj) zapewnia wykonywanie procesu, nawet gdy użytkownik wyrejestruje się z systemu (ang. log out) lub gdy nastąpi przerwanie połączenia telekomunikacyjnego z terminalem, np.:

$nohup cp plik_stary plik_nowy &

Do niszczenia procesów służy polecenie **kill,** któremu należy podać argument będący znanym identyfikatorem procesu (**PID**). Przykładowo, w celu zniszczenia procesu zainicjowanego poleceniem **sh,** zidentyfikowanego poleceniem **ps** powyżej, należy wydać polecenie:

$kill 2110

* * *

W jądrze systemu rezyduje stale proces zajmujący się nadzorowaniem pamięci operacyjnej, przydzielaniem pamięci innym procesom, przesyłaniem obrazów procesów na dysk lub inną szybką pamięć zewnętrzną oraz sprowadzaniem ich z powrotem do pamięci operacyjnej.

Niestety, z powodu braku miejsca musimy na tym zakończyć opis jądra i całego Unixa. Najwierniejszych Czytelników zapraszamy do lektury „Informatyki", gdzie zamierzamy nadal rozwijać tę tematykę na przykładzie systemu Xenix.

ROMAN GRABOWICZ
JANUSZ ZALEWSKI

Errata do artykułów: CZYM JEST UNIX?

Zeszyt 1, str. 22
pierwsza szpalta, wiersz 34 od gory
zamiast: prog. 2c
powinno być: **prog2c.**
Zeszyt 1, str. 22
druga szpalta, wiersz 12 od dołu
zamiast: -rx--------
powinno być: **-r-x------**
Zeszyt 2, str. 19, wiersz 5 od końca artykułu:
zamiast: $cat pierwsza_ liga
powinno być: **$cat > pierwsza_liga**
Zeszyt 3, str. 9, wiersz 12 od końca artykułu:
zamiast: wprowadzenie
powinno być: **wyprowadzanie**
Zeszyt 3, str. 9
pierwsza szpalta, wiersz 24 od dołu
zamiast: cd
powinno być: **cp**

Rozszerzenie pamięci mikrokomputera ZX SPECTRUM

W standardowym ZX Spectrum 48 KB zastosowano kostki 4532 o pojemności 32 Kb jako pamięć RAM zapełniającą obszar adresowy 8000 – #FFFF (32768 – 65535). Są to w rzeczywistości odrzuty produkcyjne pamięci o pojemności 64 Kb, tzn. kostki, w których stwierdzono uszkodzenie jednej połówki pamięci. Struktury takie są konfekcjonowane i oznaczane jako pamięci 32 Kb, a następnie sprzedawane po niższych cenach. Wynika z tego możliwość użycia zamiast nich pełnosprawnych pamięci 64 Kb, z tym że 32 Kb pamięci będzie nieużywane (sytuacje takie bardzo często spotyka się w komputerach z rozszerzoną pamięcią, tzn. wersjach 16 KB przerabianych na 48 KB). W takim wypadku myśl wykorzystania nieużywanej pamięci nasuwa się sama.

Propozycje rozwiązań konstrukcyjnych zwane szumnie ZX Spectrum 80 KB można było spotkać powszechnie. Różniły się one od siebie w zasadzie jedynie sposobem przełączania dodatkowych 32 KB. Wspólną ich wadą była trudność praktycznego wykorzystania dodatkowej pamięci.

Architektura mikroprocesora Z80 (zastosowanego w ZX Spectrum) umożliwia jednoczesne adresowanie do 64 KB pamięci; wynika z tego konieczność przełączania najwyższych 32 KB. Rozważmy przydatność praktyczną takiego rozwiązania. Można w nim dostrzec następujące niedogodności:

- **niedostępność pod BASIC.** Jeżeli RAMTOP ustawiony będzie powyżej adresu #7FFF (32767) wymiana obszarów 32 KB spowoduje utratę danych systemowych (m.in. stosu maszynowego), co najczęściej sprowadza się do wyzerowania systemu. Należy podkreślić, że współpraca z pewnymi programami (np. BETA BASIC) będzie niemożliwa, nawet gdy RAMTOP będzie ustawiony poniżej adresu #7FFF;
- **nieużyteczność dla asemblera.** Wykorzystanie dodatkowej pamięci dla pamiętania tekstu źródłowego (mały obszar pamięci daje się we znaki przy dłuższych programach) jest niemożliwe bez przeróbki programu asemblera.

Rozwiązanie przedstawione poniżej wykorzystuje dodatkowo 32 KB pamięci jako tzw. RAM DISC. Bodźcem do wykonania projektu był fakt, że ZX Spectrum sprawdza pamięć przy inicjalizacji systemu, niszcząc przy okazji jej zawartość. Uruchomienie zasemblowanego programu zawierającego błędy może się skończyć zawieszeniem systemu i w konsekwencji koniecznością wpisania z taśmy magnetofonowej asemblera i tekstu źródłowego. Istnienie obszaru pamięci nieniszczonego przy inicjalizacji systemu umożliwia przechowanie informacji w bezpieczny sposób, a odzyskanie jej trwa bardzo krótko.

Mapa pamięci proponowanego rozszerzenia przedstawiona została na rys. 1. Obszary pamięci 32 KB, nazwane umownie bankiem 1 i bankiem 2, pełnią rolę obszarów: roboczego (0) i przechowującego (1). Aby ułatwić gospodarkę bankami pamięci w komputerze zainstalowano dodatkowy obszar pamięci RAM (1 KB), umieszczony w obszarze adresowym pamięci ROM, która nie jest w pełni wykorzystana (pomiędzy ostatnim bajtem programu a tablicą wzorców znaków znajduje się ponad 1 KB niewykorzystanej przestrzeni zapełnionej wartością #FF). Opisany w dalszej części artykułu program umożliwia szybkie przepisywanie zawartości banków.

W wypadku użycia sygnału RESET zniszczona zostanie zawartość pamięci RAM 16 KB i roboczego banku pamięci. Zachowa się zawartość banku przechowującego i pamięci RAM, co umożliwi (na przykład) szybkie skopiowanie banku przechowującego do roboczego.

OPIS ZMIAN UKŁADOWYCH

Wszystkie oznaczenia elementów komputera ZX Spectrum są zgodne z oznaczeniami schematu ideowego tego komputera, opublikowanego w numerze 5 miesięcznika „Informatyka" z 1985 roku (str. 14 i 15).

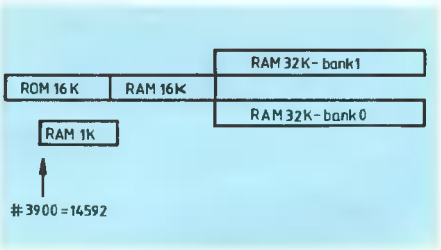
Układ sterowania pamięci

Schemat ideowy układu jest przedstawiony na rys. 2. Dekoder, skonstruowany przy użyciu bramek MCY 74023 i MCY 74001, wykrywa jednoczesne pojawienie się na młodszym bajcie szyny adresowej samych jedynek (#FF) i na szynie sterującej niskich poziomów sygnałów IORQ i WR (co odpowiada wykonywaniu przez mikroprocesor Z80 instrukcji OUT). Wyjście dekodera steruje przerzutnikiem D (MCY 74013), ktory zapamiętuje stan najmłodszego bitu szyny danych. Wyjście Q przerzutnika jest podłączone do najstarszego bitu adresowego pamięci 64 KB (przez multiplekser IC 26), co powoduje – w zależności od jego stanu – wybieranie górnej lub dolnej połówki tej pamięci.

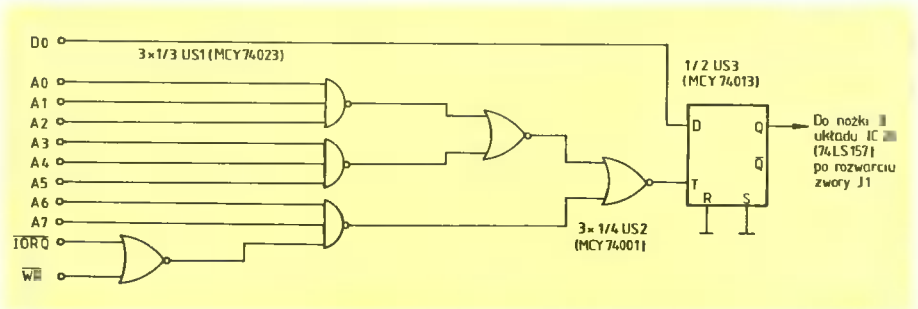
Układ RESET

Układ RESET komputera powinien spełniać następujące założenia:

- zapewniać automatyczny RESET po włączeniu komputera do sieci;
- nie powodować uszkodzenia zawartości pamięci. Punkt ten wymaga nieco szerszego omówienia. Powody uszkodzenia zawartości pamięci przez układ RESET mogą być dwojakiej natury. Pierwszym z nich jest konieczność odświeżania zawartości pamięci dynamicznych RAM. Producent gwarantuje prawidłową pracę pamięci przy cyklu odświeżania krótszym od 2 ms, co automatycznie limituje czas trwania impulsu. Drugim powodem jest właściwość mikroprocesora Z80 polegająca na tym, że gdy sygnał RESET pojawi się podczas taktu T2 lub T4, wówczas sygnał MREQ przybiera mniej więcej trzy takty później stan nieokreślony na okres jednego taktu. Może to spowodować uszkodzenie zawartości pamięci. Zjawisku temu zapobiega bramkowanie sygnału RESET opadającym zboczem sygnału M1.



Rys. 1. Mapa pamięci ZX Spectrum 81 KB



Rys. 2. Układ sterowania bankami pamięci

Schemat ideowy układu RESET jest przedstawiony na rys. 3. Generator astabilny CMOS typu MCY 74047 generuje falę prostokątną, którą jest synchronizowany licznik synchroniczny zbudowany z przerzutnikow

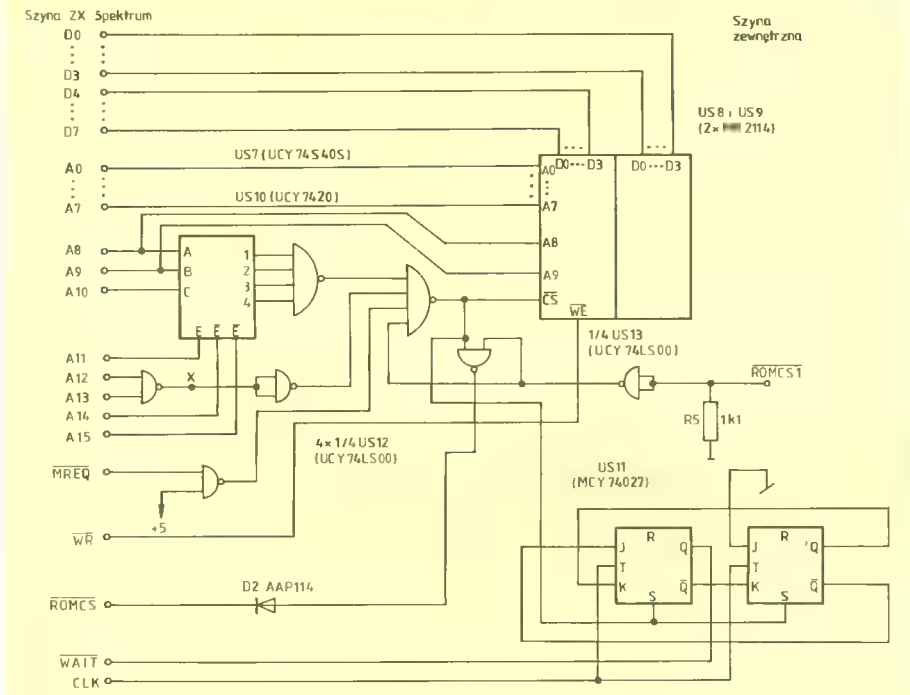
JK (MCY 74027). Licznik został zaprojektowany, tak aby jego cykl pracy sprowadzał się do generacji impulsu o czasie trwania równym dwóm okresom przebiegu generatora. Sygnał ten, po synchronizacji z przebiegiem M1, realizowanej przez przerzutnik D (MCY 74013), jest podawany na wejście RESET mikroprocesora. Wyzwalanie licznika przyciskiem RESET i automatyczny start po włączeniu komputera do sieci zapewnia układ zbudowany na bramkach MCY 74011.
UWAGA! Przed dołączeniem układu z komputera należy usunąć kondensator C27 (o pojemności 1 mikrofarada). Pełni on funkcję opóźniania sygnału RESET przy autostarcie. Pozostawienie go spowodowałoby nieprawidłową pracę układu.

Układ dodatkowej pamięci statycznej

Dodatkowa pamięć statyczna (rys. 4) została zrealizowana na dwóch kostkach MM 2114. Dekoder adresu jest zbudowany przy użyciu kostki UCY 74S405. Dekoduje on następujące stany starszego bajtu szyny adresowej #39, #3A, #3B, #3C. Po zdekodowaniu jednego z nich następuje uaktywnienie pamięci MM2114, przy jednoczesnym zablokowaniu pamięci wewnętrznej ROM. Wyprowadzone wejście ROMCS1 umożliwia blokowanie pamięci ROM niezależnie od układu i korzystanie z innych urządzeń zewnętrznych.

Ponieważ pamięci typu 2114 mają różny czas dostępu (w zależności od producenta i wersji), dodano do układu blok opóźniający zrealizowany na przerzutnikach JK typu MCY 74027. Przy stosowaniu szybkich pamięci blok ten można pominąć.

Celowe wydaje się wyposażenie układu w odłącznik. Można to zrealizować np. przez przerwanie połączenia w punkcie X (rys. 4). Powoduje to unieruchomienie dekodera (aktywny ROM) przy jednoczesnym zachowaniu aktywności wejścia ROMS1.



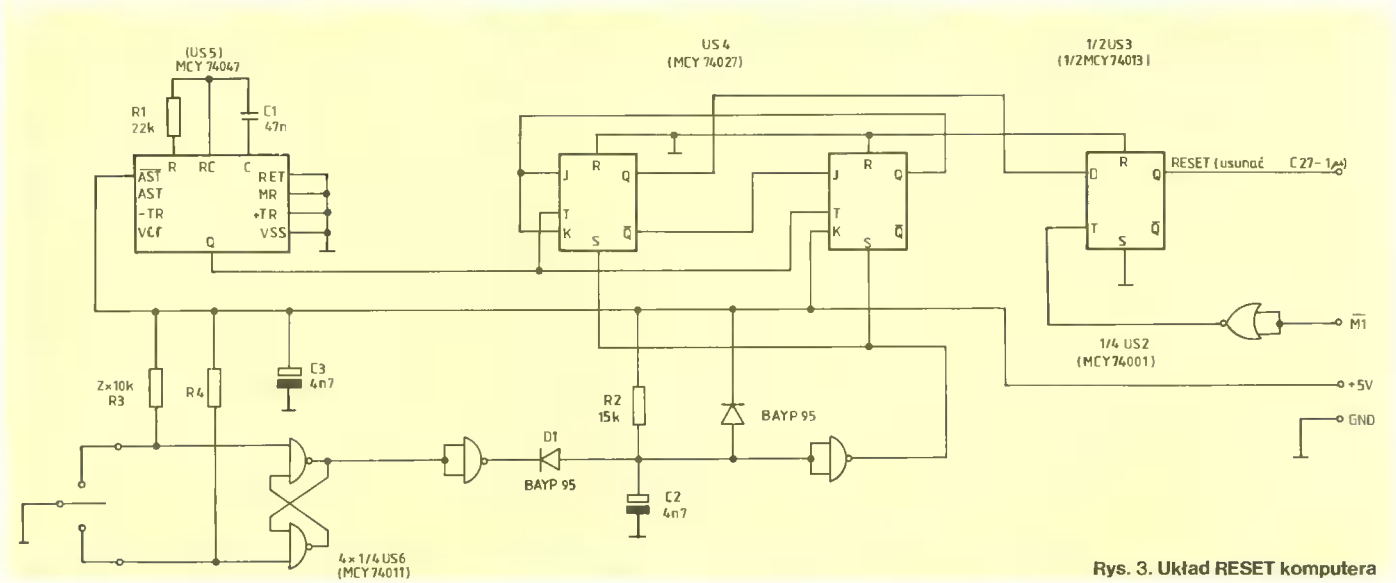
Rys. 4. Układ rozszerzenia pamięci o 1 KB

OPIS PROGRAMU

Zamieszczony program daje użytkownikowi do dyspozycji sześć opcji, których skrócone opisy wyświetlane są na ekranie przy każdym powrocie do menu. Opisy szczegółowe zamieszczone są poniżej.

Wywołanie każdej opcji odbywa się przez naciśnięcie klawisza z pierwszą literą nazwy opcji.

- **Hide** (schowaj) – powoduje przepisanie zawartości banku 0 (roboczego) do banku 1 (pamięciowego). Po wykonaniu operacji program zgłasza się z raportem „Hidden" (schowano).
- **Load** (załaduj) – powoduje przepisanie zawartości banku 1 (pamięciowego) do banku 0 (roboczego). Po wykonaniu operacji program zgłasza się z raportem „Loaded" (załadowano).
- **Swap** (zamień) – powoduje wymianę zawartości banków 0 i 1 (roboczego i pamięciowego). Po wykonaniu operacji program



Rys. 3. Układ RESET komputera

HEKSADECYMALNY WYPIS PROGRAMU

```
14592  3A 52 39 FE AA F5 C4 6B 0D F1 C2 54 39 3E 02 CD
14608  01 16 CD 34 3B C0 CD 6B 0D 3E 02 CD 01 16 01 21
14624  00 11 31 39 CD 3C 20 CD AA 3A AF 32 52 39 C3 54
14640  39 16 00 00 46 69 72 73 74 20 65 6E 74 65 72 20
14656  2D 20 69 6E 69 74 69 61 6C 69 7A 61 74 69 6F 6E
14672  0D 0D AA 15 CD B2 39 CD 3D 3A CD D6 3B CD A4 39
14688  F5 CD 6B 0D F1 21 54 39 E5 FE 4B CA 4C 3B FE 68
14704  CA 4C 3B FE 4C CA 6C 3B FE 6C CA 6C 3B FE 53 CA
14720  13 3C FE 73 CA 13 3C FE 45 CA 98 3B FE 65 CA 98
14736  3B FE 49 CA AA 3A FE 69 CA AA 3A FE 52 28 02 FE
14752  72 C0 E1 C9 76 21 3B 5C CB 6E 28 F8 CB AE 3A 08
14768  5C C9 3E 02 CD 01 16 01 7C 00 11 C1 39 CD 3C 20
14784  C9 16 06 00 13 01 4B 69 64 65 20 28 62 61 6E 6B
14800  20 30 20 74 6F 20 62 61 6E 6B 20 31 29 0D 0D 4C
14816  6F 61 64 20 28 62 61 6E 6B 20 31 20 74 6F 20 62
14832  61 6E 6B 20 30 29 0D 0D 53 77 6F 70 20 63 6F 6E
14848  74 65 6E 74 73 20 6F 66 20 62 61 6E 6B 73 0D 0D
14864  45 72 61 73 65 20 70 72 6F 67 72 61 6D 0D 0D 49
14880  6E 69 74 69 61 6C 69 7A 65 20 61 6E 64 20 63 68
14896  65 63 6B 0D 0D 52 65 74 75 72 6E 13 00 01 0E 00
14912  11 5A 3A CD 3C 20 ED 4B B2 5C CD 2B 2D CD E3 2D
14928  01 04 00 11 68 3A CD 3C 20 C9 16 00 14 14 01 13
14944  01 52 41 4D 54 4F 50 20 14 00 13 00 3E AA CD 78
14960  3A C0 3E 55 CD 78 3A C9 DD 46 00 DD 4E 01 DD 56
14976  02 DD 77 00 DD 77 02 2F DD 77 01 DD BE 01 20 10
14992  2F DD BE 00 18 05 00 00 C3 00 39 20 03 DD BE 02
15008  DD 70 00 DD 71 01 DD 72 02 C9 CD 34 3B C0 F3 01
15024  10 00 11 09 3B CD 3C 20 21 02 00 DD 21 00 40 CD
15040  6C 3A 20 55 DD 23 23 CB 74 28 F4 3E 01 D3 FF CD
15056  E9 3A 20 45 D3 FF CD E9 3A 20 3E CD 6B 0D 01 0C
15072  00 11 FD 3A CD 3C 20 FB C9 21 02 00 DD 21 00 B0
15088  CD 6C 3A C0 DD 23 23 CB 7C 28 F5 97 C9 0D 52 41
15104  4D 20 63 6B 65 63 6B 65 64 12 01 63 6B 65 63 6B
15120  69 6E 67 20 52 41 4D 12 00 CD DB 3A 01 07 00 11
15136  E4 14 CD 3C 20 3E 64 D7 C9 D3 FF 46 2F D3 FF 70
15152  23 7C B5 C9 3A B3 5C CB 7F CB F5 3E 0D D7 01 0D

15168  00 11 DE 14 CD 3C 20 3E 64 D7 F1 C9 CD 34 3B C0
15184  21 00 B0 AF CD 29 3B 20 FA 01 06 00 11 66 3B CD
15200  3C 20 AF D3 FF C9 4B 69 64 64 65 6E CD 34 3B C0
15216  21 00 B0 3E FF CD 29 3B 20 F9 01 06 00 11 84 3B
15232  CD 3C 20 C9 4C 6F 61 64 65 64 3E FF 77 23 0B 79
15248  B0 20 F7 C9 CD 6B 0D C9 01 15 00 11 C1 3B CD 3C
15264  20 CD A4 39 FE 59 28 02 FE 79 20 E8 E1 01 0A 00
15280  11 E0 57 21 BA 3B ED B0 21 00 39 01 00 04 C3 E0
15296  57 0D 41 72 65 20 79 6F 75 20 73 75 72 65 20 3F
15312  20 28 79 2F 2A 29 01 33 00 11 E0 3B CD 3C 20 C9
15328  16 00 00 11 05 52 41 4D 20 62 61 6E 6B 73 0D 6F
15344  70 65 72 61 74 69 6E 67 0D 73 79 73 74 65 6D 20
15360  20 20 0D 0D 7F 20 44 2E 41 2E 50 2E 20 31 39 38
15376  36 11 07 21 00 B0 AF D3 FF 46 2F D3 FF 4E 70 2F
15392  D3 FF 71 23 7C B5 20 EE 01 07 00 11 32 3C CD 3C
15408  20 C9 53 77 6F 70 70 65 64
```

zgłasza się z raportem „Swapped" (wymieniono).

■ **Erase** (wymaż) – powoduje zapełnienie całej dodatkowej pamięci statycznej wartością # FF (255) i powrót do systemu ZX Spectrum. Implementację tej opcji spowodował fakt, że niektóre programy fabryczne wykorzystują obszar pamięci ROM powyżej adresu #3900 (14595) jako wzorzec i inna niż #FF jego zawartość powoduje błędną pracę.

Program upewnia się ▫ nieprzypadkowości tej decyzji komunikatem „Are you sure?" (na pewno?). Wciśnięcie dowolnego klawisza (innego niż Y) spowoduje niewykonanie opcji i powrót do menu.

■ **Initialize and chek** (inicjalizacja i sprawdzenie pamięci) – powoduje sprawdzenie całej pamięci dynamicznej RAM (bez uszkodzenia jej zawartości) i ustawienie banku 0 jako roboczego. Podczas testowania pamięci jest wyświetlany komunikat „checking RAM" (pamięć RAM jest testowana). Po zakończeniu testu, program zgłasza się z raportem „RAM checked" (pamięć sprawdzona). W wypadku, gdy wykryty zostanie błąd, raport ma treść „RAM checked no good" (niedobra).

■ **Return** (powrót) – powoduje powrót do systemu ZX Spectrum bez żadnych zmian w programie lub bankach.

* * *

Zamieszczony wydruk heksadecymalny programu umożliwia wprowadzenie go do pamięci komputera. Program uruchamia się dowolną instrukcją z argumentem USR 15000 (najlepiej RANDOMIZE). Po jej wykonaniu program zgłosi się z menu lub (o ile wywołanie miało miejsce pierwszy raz) zrealizuje automatyczną inicjalizację. Z tego powodu przy pierwszym wejściu jest testowana wartość RAMTOP i ▫ ile jest większa od 32767, program wraca do systemu z komunikatem „RAMTOP no good" (niedobra wartość RAMTOP). Z tego powodu należy zapamiętać na taśmie wpisany z listingu program przed jego uruchomieniem – nawet jeżeli był wpisany bezbłędnie, jego pierwsze uruchomienie spowoduje jego samomodyfikację.

W prawym górnym rogu ekranu wyświetlana jest aktualna wartość RAMTOP, co może być przydatne przy pracy z asemblerem. Program wypisuje raporty w najniższych liniach ekranu, normalnie zarezerwowanych dla komunikacji z systemem.

Na zakończenie należy dodać, że niektóre urządzenia zewnętrzne wykorzystują zajmowany przez program obszar adresowy, umieszczając tam własne oprogramowanie, co może uniemożliwić właściwą pracę programu.

KRZYSZTOF AMBORSKI
DARIUSZ ADAM PRZYGODA

BASIC dla początkujących

W pierwszych czterech odcinkach naszego kursu nauczyliśmy się już pewnych podstawowych zasad programowania w języku BASIC komputera Commodore 64, a także Spectrum. Dotychczas zajmowaliśmy się głównie zmiennymi liczbowymi. Teraz pokażemy, jak można przetwarzać litery oraz pozostałe rodzaje znaków (przestankowe, graficzne), a więc jak operować ciągami znaków.

Jeśli po raz pierwszy zetknąłeś się z naszym kursem, przeczytaj zamieszczone w ramce powtórzenie wiadomości z poprzednich odcinkow. Jeśli czytałes poprzednie odcinki, to powtórzenie wiadomości też Ci nie zaszkodzi. Po tej wstępnej rozgrzewce możemy przystąpić do dalszej nauki.

```
10    PRINT "A  B  OBW.  POW.  PRZEK."
20    FOR A=1 TO 20
30    FOR B=1 TO 20
40    GOSUB 100
50    FOR WA=1 TO 500
60    NEXT WA
70    NEXT B
80    NEXT A
90    END
100   O=2*A+2*B
110   PO=A*B
120   PR=SQR(A*A+B*B)
130   GOSUB 200
140   PRINT A;B;O;PO;PR
150   RETURN
200   PR=INT(PR*100+.5)/100
210   RETURN
```

Wydruk 1. Rozwiązanie ćwiczenia z poprzedniego odcinka kursu

Na wydruku 1 pokazano rozwiązanie zadania domowego z czwartego odcinka. Nie jest to jednak rozwiązanie idealne – wyświetlane liczby nie są uporządkowane w postaci przejrzystej tabeli. Po dzisiejszej porcji nowych wiadomości będziesz już wiedzieć, jak takie liczby uporządkować, aby otrzymać estetyczne i przejrzyste tabele.

Najpierw zajmiemy się problemem, który można by nazwać drukowaniem niczego. Wprowadź:

```
?
```

i wciśnij klawisz RETURN. Pomiędzy wprowadzonym przez Ciebie znakiem zapytania a wysłanym przez komputer komunikatem READY znajdują się tylko dwa znaki odstępu (spacje). Wprowadź teraz:

```
?" "
```

i ponownie wciśnij klawisz RETURN. Wynik jest identyczny! W pierwszym wypadku komputer rzeczywiście nic nie wyprowadzał na ekran, lecz w drugim – wyprowadził znak spacji. Nie dało się to jednak stwierdzić na ekranie. Wprowadź teraz:

```
A$="":?A$
```

i wciśnij klawisz RETURN. Rezultat jest w dalszym ciągu taki sam, tzn. żaden. Wprowadź jeszcze:

```
A$=" "?A$
```

Także i tym razem nic się nie zmieniło.

Długość ciągu znaków

Poznamy teraz pierwszą funkcję działającą na ciągach znaków: LEN(zmienna). LEN jest skrótem od angielskiego słowa length (długość). Za pomocą funkcji LEN można więc otrzymać liczbę znaków wchodzących w skład ciągu. Wprowadź kolejno:

```
A$="":?LEN(A$)
A$=" ":?LEN(A$)
A$="MIKROKLAN":?LEN(A$)
```

Kolejno pojawiły się liczby 0, 1 i 9. Są to rzeczywiście długości ciągów znaków znajdujących się w momencie wyprowadzania w zmiennej A$

Zapamiętaj. Funkcja LEN(zmienna) może być wykorzystana do uzyskania liczby znaków składających się na ciąg przechowywany w danej zmiennej.
Wprowadź teraz:

```
A$="Jan "
B$="Kowalski"
```

i wciśnij klawisz RETURN. W komputerze zapamiętane zostały ciągi znaków. Możesz je łatwo obejrzeć, wprowadzając:

```
?A$, B$
```

Aby ten napis miał lepszy wygląd, trzeba wprowadzić:

```
?A$;B$
```

Widać teraz, że wprowadzenie znaku spacji jako ostatniego ze znaków zapamiętanych w zmiennej A$ było w pełni usprawiedliwione. Bez tego nazwisko i imię stanowiłyby jeden wyraz.

Łączenie ciągów znaków

Ciągi znaków przechowywane w zmiennych A$ i B$ można ze sobą połączyć i utworzyć w ten sposób jeden ciąg. Wprowadź w tym celu:

```
C$=A$+B$:?C$
```

Na ekranie pojawił się identyczny napis, jak w poprzednim przykładzie.

Zapamiętaj. Za pomocą znaku '+' można łączyć ze sobą ciągi znaków.

Fragmenty ciągów znaków

Załóżmy, że zmienne A$ i B$ już nie istnieją, natomiast istnieje jeszcze zmienna C$, i że na podstawie jej zawartosci chcemy odtworzyć poprzednie wartości zmiennych A$ i B$. Wartości te umieścimy w zmiennych D$ i E$. Wprowadź:

```
D$=LEFT$ (C$,4)
E$=RIGHT$ (C$,8)
```

i wyswietl rezultat:

```
?D$
?E$
```

Otrzymaliśmy ponownie oddzielnie imię i nazwisko.

Zapamiętaj. Za pomocą funkcji LEFT$ (zmienna, długość) można otrzymać znajdującą się z lewej strony część ciągu znaków. Pierwszy parametr jest nazwą zmiennej zawierającej cały ciąg znaków. Drugi parametr mówi, ile znaków, począwszy od pierwszego z lewej, chcemy uzyskać.

Zapamiętaj. Za pomocą funkcji RIGHT$(zmienna, długość) można otrzymać znajdującą się z prawej strony część ciągu znaków. Pierwszy parametr to nazwa zmiennej, drugi – liczba znaków, jaką chcemy uzyskać, licząc od prawej strony.
Wprowadź teraz dla odmiany:

```
FOR I=1 TO LEN (C$):
?MID$ (C$, I, 1):NEXT
```

W wyniku otrzymaliśmy imię i nazwisko Jana Kowalskiego wyświetlone pionowo: każdy wiersz rozpoczyna się od kolejnej litery i składa tylko z niej. Funkcja LEN(zmienna) została tu użyta jako parametr pętli (wartość końcowa). Pamiętaj jednak, że każda pętla wykona się przynajmniej jeden raz. Gdyby ciąg znaków zawarty w zmiennej C$ miał długosc 0, doprowadziłoby to do błędu wewnątrz pętli.

Zapamiętaj. Za pomoca funkcji MID$ (zmienna, numerznaku, długość) można wycinać znaki ze środka ciągu znaków. Numer

znaku mówi, od którego miejsca zostaną one pobrane. Długość oznacza liczbę pobranych znaków. Jeśli pominie się ten ostatni parametr, zostaną pobrane wszystkie znaki pozostałe do końca ciągu.

Zastąp teraz wewnątrz pętli FOR...NEXT zmienną C$ przez 'A$+B$' – wynik jest identyczny.

Zapamiętaj. Zamiast nazw zmiennych w funkcjach LEFT$, RIGHT$ i MID$ można użyć wyrażeń złożonych z nazw zmiennych tekstowych i samych ciągów znaków połączonych ze sobą znakami '+'

Zapamiętaj. W funkcjach LEFT$, RIGHT$, MID$ parametry liczbowe (czyli ostatni parametr w LEFT$ i RIGHT$ oraz dwa ostatnie parametry w MID$) mogą być także zmiennymi liczbowymi lub wyrażeniami liczbowymi wyliczanymi na podstawie wielu zmiennych liczbowych.

Prawdopodobnie wydaje Ci się to wszystko nieco skomplikowane. Nie martw się – gdy chwilę poeksperymentujesz, wszystko stanie się proste i oczywiste.

Przedstawimy teraz w miarę wyczerpujący przykład. Otóż chcemy rozdzielić imię od nazwiska, nie znając ich dokładnie, lecz wiedząc, że są zapamiętane w zmiennej C$ i że pomiędzy imieniem i nazwiskiem znajduje się spacja (znak odstępu). Program realizujący to zadanie jest przedstawiony na wydruku 2.

Na początku trzeba oczywiście ustalić, na której pozycji występuje rozdzielająca spacja. W tym celu przeszukujemy w jednej pętli całą zmienną, przerywając poszukiwania w chwili, gdy napotkamy znak spacji. Długość całego ciągu (imienia, spacji i nazwiska) zapamiętaliśmy wcześniej w zmiennej L. W wierszu 130 znajduje się polecenie skoku, który jest wykonywany w wypadku znalezienia znaku spacji. Następuje wtedy wyjście z pętli poszukiwania.

Poeksperymentuj nieco z tym programem, zmieniając imię i nazwisko w wierszu 10.

```
10    C$="JAN KOWALSKI"
20    REM POSZUKIWANIE SPACJI
30    L=LEN(C$)
40    FOR I=1 TO L
50    IF MID$(C$,I,1)=" " THEN 300
60    NEXT
70    PRINT "BRAK SPACJI"
80    END
100   REM ROZDZIELENIE ZA POMOCA LEFT$ I MID$
110   A$=LEFT$(C$,I)
120   B$=MID$(C$,I+1)
130   PRINT A$
140   PRINT B$
150   END
300   REM ROZDZIELENIE ZA POMOCA LEFT$ I RIGHT$
310   A$=LEFT$(C$,I)
320   B$=RIGHT$(C$,L-I)
330   PRINT A$
340   PRINT B$
350   END
```

Wydruk 2. Rozdzielenie ciągu znaków

Zmień także w wierszu 50 numer wiersza po słowie THEN z 300 na 100. Co się wtedy stanie?

Otóż po zmianie numeru wiersza po słowie THEN w wypadku znalezienia spacji był wykonywany podprogram rozpoczynający się w wierszu 100. Rozdzielenie wykonano tu za pomocą funkcji LEFT$ i MID$. Lewą część ciągu łącznie ze spacją umieszczono w zmiennej A$, a prawą – w B$. W funkcji MID$ pominięto ostatni parametr.

Przed zmianą numeru wiersza za słowem THEN po znalezieniu spacji był wykonywany podprogram 300. Zamiast funkcji MID$ do uzyskania nazwiska użyto tu funkcji RIGHT$. Było to nieco bardziej skomplikowane niż w poprzednim wypadku, gdyż trzeba było obliczyć, ile znaków należy wyciąć z prawej strony. Wykonano to odejmując od całkowitej długości ciągu (zmienna L) liczbę znakow wyciętych przez funkcję LEFT$(zmienna I).

Przekształcenia

Przekonaliśmy się już, że pomiędzy liczbami a ciągami znaków (oraz zmiennymi liczbowymi a zmiennymi tekstowymi) istnieje podstawowa różnica. Pokażemy to jeszcze na dwóch krótkich przykładach:

```
10 A=123
20 B=456
30 C=A+B
40 ?C
```

Po wykonaniu programu na ekranie pojawi się wynik dodawania arytmetycznego: '579' Dodaj teraz do nazw zmiennych znak $, a z

```
100 A=.82
110 FORI=1 TO
120 A=A*8
130 GOSUB 200
140 NEXT
150 END
200 A=INT(100*A+.5)/100
210 A$=STR$(A)
220 L=LEN(A$)
230 IF L<3 GOTO 260
240 IF MID$(A$,L-2,1)="." THEN A$=LEFT$(A$,L-3)+","+RIGHT$(A$,2)   GOTO 270
250 IF MID$(A$,L-1,1)="." THEN A$=LEFT$(A$,L-2)+","+RIGHT$(A$,1)+"0"   GOTO 270
260 A$=A$+",00"
270 L=LEN(A$)
280 IF A>999.99 THEN A$=LEFT$(A$,L-6)+"."+RIGHT$(A$,6)
290 IF A>999999.99 THEN A$=LEFT$(A$,L-9)+"."+RIGHT$(A$,10)
300 A$=RIGHT$("                "+A$,14)
310 PRINTA$,A
320 RETURN
```

Wydruk 3

Powtórzenie wiadomości z części 1-4

- Za pomocą jednego polecenia PRINT można wyprowadzić zmienne i stałe, zarówno liczbowe jak i znakowe (tekstowe).
- Polecenie GOTO służy do zmiany normalnego przebiegu programu. Za słowami GOTO należy podać numer wiersza, do którego nastąpi skok.
- Każda pętla programowa zostanie wykonana przynajmniej jeden raz.
- Po wyjściu z pętli programowej zmienna sterująca pętli ma wartość równą wartości końcowej zwiększonej o 1.
- Wartości zmiennych mogą być wyświetlone w trybie bezpośrednim za pomocą zwykłego polecenia PRINT.
- Interpreter języka BASIC umożliwia w dowolnej chwili zatrzymanie programu i obejrzenie wartości zmiennych.
- Funkcja INT () może służyć do obcięcia cyfr po kropce dziesiętnej. Wynikiem jest liczba całkowita. Liczby całkowite nie są zaokrąglane.
- W nawiasach po słowie INT mogą znajdować się wyrażenia arytmetyczne zbudowane z liczb, zmiennych i operatorów arytmetycznych.
- Jeśli argumentem funkcji INT jest liczba ujemna, to wartością funkcji będzie najbliższa liczba całkowita większa lub równa argumentowi.
- Podprogramy to fragmenty programu wywoływane za pomocą polecenia GOSUB wraz ze znajdującym się za nim numerem wiersza.
- W podprogramach zmienne mają takie wartości, jakie miały w chwili wywołania podprogramu z programu głównego.
- Parametry wejściowe i wyjściowe służą do przekazywania wartości pomiędzy programem głównym i podprogramami.
- Parametr wejściowy może być jednocześnie parametrem wyjściowym.
- Podprogramy mogą być zagnieżdżone jedne w drugich.

obu stron stałych umieść znaki " ", otrzymując:

```
10 A$="123"
20 B$="456"
30 C$=A$+B$
40 ?C$
```

Wykonaj program. W wyniku otrzymasz połączenie dwóch ciągów znaków: '123456'.

Ten sam wynik co w pierwszym przykładzie można też otrzymać po wykonaniu poniższego programu:

```
10 A$="123"
20 B$="456"
30 C=VAL (A$)+VAL(B$)
40 ?C
```

Zapamiętaj. Za pomocą funkcji VAL(zmienna tekstowa) można otrzymać wartość liczbową odpowiadającą ciągowi znaków zapamiętanemu w zmiennej tekstowej i przedstawiającemu tę liczbę.

Uwaga! Jeśli pierwszy znak w ciągu nie jest cyfrą, to uzyskana wartość jest równa 0. Jeśli po znakach przedstawiających liczbę

znajdują się inne znaki (litery lub znaki specjalne), to są one ignorowane.

Funkcja STR$ jest odwrotnością funkcji VAL. Możemy ją wykorzystać do uzyskania w inny sposób wyniku podobnego do otrzymanego w drugim przykładzie. Najpierw zmienne liczbowe przekształca się w ciągi znaków, a następnie dwa ciągi łączy się w jeden:

```
10 A=123
20 B=456
30 C$=STR$(A)+STR$(B)
40 ?C$
```

Zapamiętaj. Za pomocą funkcji STR$(zmienna liczbowa) można przekształcić liczbę w odpowiadający jej ciąg znaków. Pierwszy znak ciągu odpowiada znakowi liczby: dla liczb ujemnych jest to minus, a dla liczb dodatnich bądź znak – znak spacji.

Po wykonaniu programu otrzymujemy 123 456', a więc wynik nieco różniący się od otrzymanego w drugim przkładzie. Puste miejsca odpowiadają bowiem znakom liczb, które weszły w skład ciągu

Jak uzyskać przejrzysty katalog cenowy

Na podstawie posiadanych wiadomości możemy się teraz pokusić o zorganizowanie wyświetlania cen w postaci katalogu. Oznacza to, że kropkę dziesiętną używaną w krajach anglosaskich zastąpimy używanym u nas przecinkiem, zaś za pomocą kropki zaznaczymy pozycje tysięcy. Postaramy się też o to, aby przecinek dziesiętny zawsze znajdował się na tej samej pozycji.

Omawiany program przedstawiony jest na wydruku 3. Program główny zawarty jest w wierszach 100-140. W każdym przebiegu pętli generowana jest pewna liczba, po czym następuje wywołanie podprogramu 200, w ktorym liczba ta ulega przekształceniu i wyświetleniu. Przyjrzyjmy się bliżej temu podprogramowi.

W wierszu 200 następuje zaokrąglenie liczby w sposób uwzględniający dwie cyfry dziesiętne po przecinku. W wierszu 230 bada się, czy po kropce dziesiętnej są jakieś cyfry. Jeśli nie, to trzeba dodać dwa zera (wiersz 260). Jeśli jest tylko jedna cyfra po kropce, to trzeba dodać jedno zero (wiersz 250). Ponadto w każdym wypadku kropka dziesiętna jest zastępowana przez przecinek. Na przykład w wierszu 240 wygląda to następująco. Najpierw sprawdza się, czy na pozycji trzeciej od końca była kropka. Jeśli tak, to za pomocą funkcji LEFT$ wycina się część całkowitą ceny (z lewej strony ciągu, od pierwszego znaku aż do kropki, ale bez niej). Do wycięcia części ułamkowej ceny (w tym wypadku dwóch cyfr) służy funkcja RIGHT$. Obie części łączy się następnie razem, wstawiając pomiędzy nie przecinek, za pomocą operatorów '+'.

Na skutek tych operacji mogła ulec zmianie długość ciągu w zmiennej A$. Dlatego w wierszu 270 długość ta jest ustalana na nowo. W wierszu 280 sprawdza się, czy liczba zawiera cyfry przekraczające pozycję tysięcy. Jeśli tak, ciąg znowu ulega rozdzieleniu i jest wstawiana kropka oddzielająca liczbę tysięcy. Podobnie w wierszu 290 ewentualnie jest wstawiana kropka oddzielająca liczbę milionów. Liczby będące parametrami funkcji LEFT$, RIGHT$ i MID$ otrzymano po przeanalizowaniu problemu. Warto, abyś na kilku przykładach sprawdził za pomocą ołówka i kartki papieru ich poprawność. Dla konkretnych liczb musiałbyś w takim wypadku policzyć znaki od lewej do prawej strony.

Aby wyniki wydrukować w postaci tabelarycznej, trzeba zapewnić położenie przecinka dziesiętnego zawsze w tej samej kolumnie ekranu. W tym celu do uzyskanego ciągu znakow dodaje się na początku dostatecznie dużo spacji (w naszym wypadku 14) i z tak powstałego ciągu wycina się 14 znaków położonych z prawej strony. Odbywa się to w wierszu 300, zaś w wierszu 310 następuje wyświetlenie wyniku (dla porównania także i liczby w normalnej postaci).

Z pewnością jest to najbardziej skomplikowany z dotychczas przedstawionych przykładów. Warto jednak włożyć nawet sporo wysiłku, aby go dokładnie zrozumieć

Na zakończenie proponujemy Ci rozwiązanie następującego zadania: ćwiczenie zamieszczone w poprzednim zeszycie, a przedstawione w obecnym odcinku na wydruku 1, należy przekształcić do takiej postaci, aby wszystkie dane były wyświetlane w uporządkowanej, tabelarycznej postaci.

micro

DLA UŻYTKOWNIKÓW ZX SPECTRUM

W ZX Spectrum nie ma instrukcji LEFT$, RIGHT$, MID$. Wycinanie fragmentów znaków jest jednak możliwe, a nawet prostsze. W tym celu z ciągu znaków dodaje się w nawiasach wyrażenie opisujące sposób przekształcenia:

ciągznaków (nrpoczątkowy TO nrkońcowy)

Wyrażenia nrpoczątkowy i nrkońcowy są numerami znaków w ciągu przekształcanym, które staną się odpowiednio znakiem początkowym i znakiem końcowym ciągu przekształconym. Pominięcie nrpoczątkowego powoduje pobieranie znaków od początku, zaś nrkońcowego – do końca ciągu przekształcanego.

A zatem użytkownicy ZX Spectrum zamiast funkcji:

LEFT$ (ciągznaków, liczba znaków)
RIGHT$ (ciągznaków, liczba znaków)
MID$ (ciągznaków, pozycja, liczba znaków)

powinni stosować odpowiednio operacje:

ciągznaków (TO liczbaznaków)
ciągznaków (LEN (ciągznaków) – liczbaznaków+1 TO)
ciągznaków (pozycja TO pozycja+liczbaznaków)

```
10  LET C$="JAN KOWALSKI"
100 REM POSZUKIWANIE SPACJI
110 LET L=LEN(C$)
120 FOR I=1 TO L
130 IF C$(I)=" " THEN GOTO 200
131 REM MOZNA BYC ROWNIEZ
132 REM IF C$(I TO I)=" " THEN GOTO 200
140 NEXT I
150 PRINT "BRAK SPACJI"
160 STOP
200 REM ROZDZIELENIE ZA POMOCA TO
210 LET A$=C$(TO I)
220 LET B$=C$(I+1 TO)
230 PRINT A$'B$
250 STOP
```

Rozdzielenie ciągu znaków w komputerze ZX Spectrum

Sprostowanie

Przepraszamy Czytelników za pominięcie wydruku 1 w artykule „Klawiatura hallotronowa" (zeszyt 3, 1986 r. str. 16). Wydruk ten przedstawiamy poniżej.

```
.M 0,FF

0000 FF FF FF 96 FF FF FF FF FF FF FF FF FF FF FF FF  ................
0010 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
0020 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
0030 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF  ................
0040 FF 00 D2 D5 DF D6 D7 DD FF F2 F7 9F 8F 90 96 8A  ................
0050 FF FF 82 84 D4 93 94 95 FF DF FF C0 83 C1 C3 92  ................
0060 FF F6 DE DD DC DD DA D9 FF E4 8E 88 9A 8D 0D 86  ................
0070 FF 81 9E 8C 08 99 98 97 FF FF 85 87 9C 89 9D 91  ................
0080 FF A0 C2 C5 CF C6 C7 C8 FF F2 F7 FF EF F0 F6 E0  ................
0090 FF FF E2 E4 C4 F3 F4 F5 FF DF FF D0 E3 D1 D3 F2  ................
00A0 FF F6 CE CD CC CB CA C9 FF E4 EE E8 FA ED EB E6  ................
00B0 FF E1 FE EC FB F9 F8 F7 FF FF E5 E7 FC E9 FD F1  ................
00C0 FF F5 C2 C5 CF C6 C7 C8 FF F2 F7 BF AF 00 B6 A0  ................
00D0 FF FF A2 A4 C4 B3 B4 B5 FF DF FF D0 A3 D1 D3 B2  ................
00E0 FF F6 CE CD CC CB CA C9 FF E4 AE A0 BA AD AB A6  ................
00F0 FF A1 BE AC BB B9 B8 B7 FF FF A5 A7 BC A9 0D B1  ................
.
```

Jakość dyskietek (2)

Po omówieniu w poprzednim numerze technologii produkcji oraz testu własności mechanicznych dyskietek, obecnie scharakteryzujemy powszechnie stosowane metody zapisu oraz wyniki testowania podstawowej właściwości elektrycznej, jaką jest poziom sygnału wyjściowego z uwzględnieniem wpływu temperatury.

Zapis na dyskietkach nie różni się zasadniczo od zapisu na taśmach magnetofonowych i wideo. Wykorzystywane są tu właściwości tlenku żelaza (lub podobnego materiału), którym powleczony jest nośnik. Jedną z własności jest zdolność zachowania przez dłuższy czas wytworzonego namagnesowania, drugą natomiast jest to, że w polu magnetycznym można rozróżnić zawsze dwa bieguny. Do zapisu cyfrowego, w którym muszą być rozróżniane jednoznacznie dwa stany, istnieją więc pełne warunki realizacji. Pojedyncza informacja (bit) powstaje na powierzchni dyskietki przez magnetyczne ukierunkowanie maleńkiego obszaru na jej powierzchni.

W odróżnieniu od zapisu dźwięku, w zapisie dyskietek stosuje się tzw. metodę nasycenia, polegającą na dobraniu takiej wartości prądu zapisu, aby została namagnesowana cała warstwa tlenku. Podczas czytania dyskietki warstwa magnetyczna dyskietki przesuwa się pod głowicę. Zmiana przepływu prądu wytwarza w cewce niewielkie napięcie rzędu kilku miliwoltów, ktore zostaje wzmocnione i przekształcone na impulsy. Impulsy te stanowią dokładne odwzorowanie zapisanych danych.

Różnorodność formatów

Powierzchnia dyskietki jest podzielona na koncentrycznie rozmieszczone ścieżki, a te z kolei – na wydzielone obszary, zwane sektorami. W najczęściej obecnie stosowanych dyskietkach o średnicy 130 mm (5,25 cala) występuje zwykle 40 lub 80 ścieżek. Każda ścieżka dzieli się na 8-20 sektorów, odpowiadających wycinkowi okręgu. Oprócz danych każdy sektor zawiera również informację wskazującą na jego położenie, a także informacje służące do synchronizacji strumienia danych oraz wykrywania błędów. Między dwoma sektorami oraz na końcu ścieżki znajdują się przerwy (ang. gaps) długości 25-50 bajtów. Mają one zapobiec powtórnemu zapisaniu kolejnego sektora w sytuacji, gdy w napędzie wystąpią nierównomierności obrotów dyskietki.

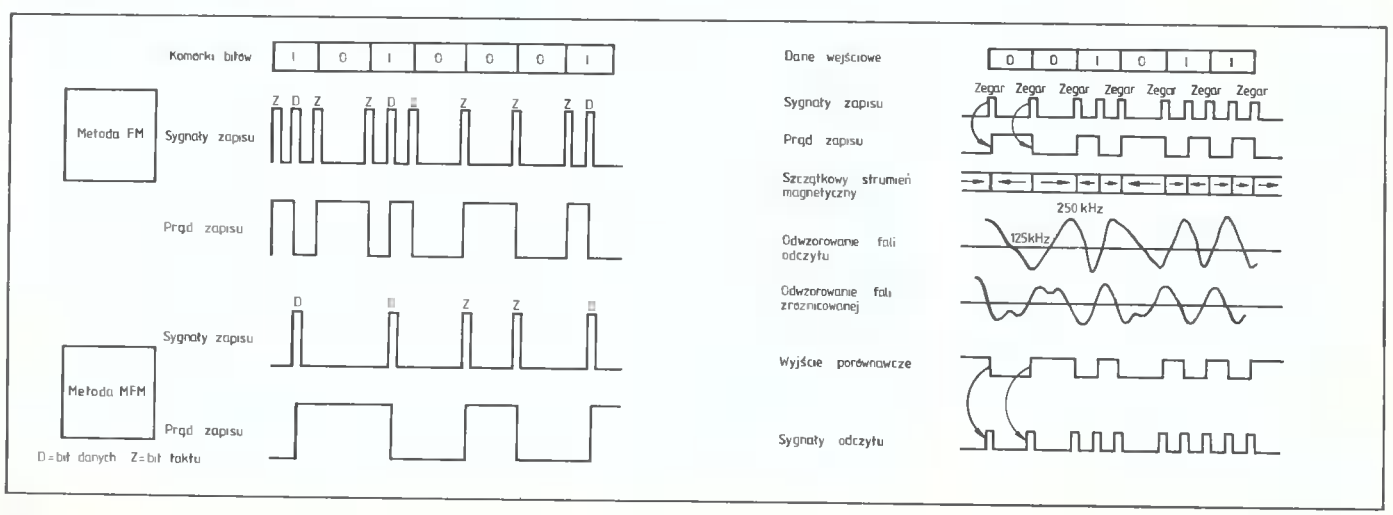
Chociaż już istnieją normy dotyczące układu danych na dyskietce oraz metody ich zapisu, to niestety stosują się do nich tylko nieliczni producenci. Wynikiem takiej sytuacji są bardzo zróżnicowane pojemności pamięci dyskietkowych. Powszechnie stosowane są bloki danych zawierające od 128 do 1024 bajtów na jeden sektor. Przykładowo mikrokomputer IBM PC jest wyposażony w napędy z zapisem obustronnym po 40 ścieżek i zapisuje na jednej ścieżce 8 lub 9 sektorów po 512 bajtów. W ten sposób pojemność dyskietki w jednym napędzie wynosi 320 lub 360 kB. Komputer Sirius 1 firmy Victor ma napędy z zapisem 80-ścieżkowym oraz zmienną liczbę sektorów na jednej ścieżce (11-19). W napędzie z zapisem dwustronnym przy identycznej wielkości bloków można zatem umieścić trzykrotnie więcej danych, a mianowicie 1,2 MB.

Najważniejsze metody zapisu

Nie tylko format, ale również metody zapisu stosowane przez poszczególnych producentów są różne. Obecnie stosowane są trzy metody: FM, MFM oraz GCR.

FM (frequency modulation – modulacja częstotliwości) jest najstarszą, a jednocześnie najbardziej niezawodną metodą zapisu na dyskietkach. Jednakże nie jest sprawą prostą (jakby się to pozornie wydawało) bezpośrednie przekształcenie bitu informacyjnego w odpowiednio ukierunkowane namagnetyzowanie. Przekształcenie takie spowodowałoby, że sektor, w którym zapamiętano 512 razy wartość zero, podczas odczytu nie wywołałby ani jednej zmiany kierunku przepływu prądu, co oznacza całkowitą niemożliwość interpretacji danych. Z tego powodu do strumienia danych są dodawane tzw. informacje taktujące. Bit danych przy metodzie FM charakteryzuje się tym, że ma stały czas trwania i rozpoczyna się zmianą kierunku przepływu prądu, który reprezentuje wspomnianą informację taktującą. Kierunek strumienia magnetycznego nie ma w tym wypadku żadnego znaczenia. Jeżeli ma być zapisany bit 0, to między taktami nie

Obraz sygnałów zapisu FM oraz MFM



dzieje się nic. Dla bitu 1 między dwoma taktami jest wprowadzana dodatkowa zmiana kierunku przepływu prądu. Wynik ten poglądowo przedstawiono na rysunku. Analizując wykres prądu zapisu można zauważyć, że w taktach mogą wystąpić tylko dwie częstotliwości – stąd nazwa modulacja częstotliwości – albo metoda dwóch częstotliwości. W dokumentach normalizacyjnych zrzeszenia europejskich producentów komputerów ECMA[1] ustalono te częstotliwosci na 62,5 oraz 125 kHz, odpowiednio do podwójnej liczby zmian przepływu na sekundę.

Okazało się, że format zapisu FM nie jest zbyt oszczędny pod względem zajmowanej powierzchni. Po pierwsze, zmiany namagnesowania nie mogą być zbyt zagęszczone, a po drugie połowę tych zmian stanowią takty, a nie dane. Stąd metoda FM umożliwia zapamiętanie tylko połowy danych, jakie teoretycznie byłyby możliwe.

Aby usunąć wspomnianą wadę opracowano metodę MFM – Modified Frequency Modulation – zmodyfikowaną modulację częstotliwości. Przypomina ona w zasadzie metodę FM, różniąc się tylko sposobem włączania taktów do strumienia danych. Polega on na eliminowaniu zbędnych zmian kierunku magnesowania. Jeden bit ma tu również stały czas trwania, a obowiązują następujące dwie zasady zmian kierunku przepływu prądu:

- zmiana następuje w środku (czasowym) bitu o wartości 1,
- zmiana następuje, jeżeli dwa kolejne bity mają wartość 0.

Na podstawie rysunku można stwierdzić, że do zapisu tych samych danych używa się tylko połowy zmian kierunku przepływu. Zachowując taką samą gęstość zmian kierunków przepływu można więc osiągnąć przy metodzie MFM w porównaniu do FM podwójną pojemność pamięci. Zaletę taką trzeba jednak okupić kilkoma istotnymi wadami. Znacznie mniejsza jest pewność zapisu danych. Przy zapisie występują trzy zamiast dwóch różnych częstotliwości, co wymaga dokładniejszej regulacji napędu. Szczególnie niebezpieczne jest zjawisko przesunięcia bitów (ang. bit shifting), występujące zwłaszcza na wewnętrznych ścieżkach dyskietki. Pojęcie to oznacza zniekształcenia sygnałów odczytu, jakie powstają wskutek zagęszczenia sekwencji zmian przepływu. Należy zwrócić uwagę na to, że szybkość napędu wynosząca 300 obrotów/min jest stała, a więc przy jednakowej pojemności jednej ścieżki, tym mniej miejsca przeznacza się dla jednego bitu, im mniejszy jest obwód ścieżki. Przesunięcia bitów można zmniejszyć przez regulację napędu, pozostanie jednak problem wynikający z oddziaływania temperatury.

[1] European Computer Manufactures Association

Dla metod FM oraz MFM istnieją gotowe układy sterujące, które znacznie uprościły projektowanie jednostek pamięci. Jest to prawdopodobnie przyczyną przyjęcia tych właśnie metod w dokumentach normalizacyjnych ANSI, DIN oraz ECMA, a nie metody GCR, choć jest ona technicznie bardziej zaawansowana.

Trzecią, równie powszechną metodą zapisu jest GCR (Group Code Recording – rejestracja z kodowaniem grup), rezerwująca dla każdego bitu miejsce o stałej długości. W zależności od długości promienia, na którym znajduje się zapisywana ścieżka, szybkość obrotów dyskietki jest oczywiście różna. Powoduje to, że poszczególne tory ścieżek mają różną liczbę sektorów.

Metoda GCR zapewnia dużą pojemność pamięci oraz pewność zapisu danych, zbliżoną do osiąganej w metodzie FM. Nie ma ona również wad metody MFM. Cechą wyróżniającą GCR jest samotaktowanie, polegające na tym, że do strumienia danych nie włącza się taktow. Cel ten osiąga się dzięki rozłożeniu każdego bajtu informacji na dwa 4-bitowe półbajty, przy czym z każdych czterech bitów jest tworzony piąty bit zgodnie z kluczem pokazanym w tabeli 1. Charakterystyczną cechą tego kodu jest, że nigdy nie powtarza się kolejnej wartości 0, co z technicznego punktu widzenia nie przedstawia większych trudności. Ta przejrzysta i znacznie prostsza w porównaniu do poprzednio omówionych metoda umożliwia zapisanie dwu i pół raza więcej danych niż metodą FM.

GCR jest obecnie stosowana w mikrokomputerach firm Commodore, Apple (Macintosh) oraz Victor, ale mimo swej wyższości technicznej nie ma perspektyw większego powodzenia na rynku. Ani komitety normalizacyjne, ani IBM jako inspirator quasi-norm, nie podejmują żadnych działań w tym kierunku.

Badania

W celu zwiększenia stopnia pewności zapisu danych, we wszystkich metodach występują działania mające na celu eliminację błędów zapisu i odczytu. Polegają one na dodawaniu do bitów danych informacji umożliwiających wykrywanie błędów. Najczęściej jest to realizowane za pomocą sumy kontrolnej, zapamiętanej w dwóch bajtach i wykorzystywanej przy późniejszym odczycie do kontroli danych. Jeżeli podczas odczytu (lub zapisu) wystąpi błąd, system operacyjny powoduje wielokrotne powtórzenie operacji. Gdy wynik tego działania będzie negatywny, pojawią się typowe komunikaty o błędach „Read fault error" (błąd wadliwego odczytu) oraz „Write fault error" (błąd wadliwego zapisu). Doświadczenia wskazują, że błędy takie występują wtedy, gdy komputer jest eksploatowany w ekstremalnych warunkach temperaturowych. Zimą w pomieszczeniach bez pełnej klimatyzacji może zdarzyć się, że rano temperatura będzie wynosić zaledwie 13°C. Aby stwierdzić, czy źródłem błędu jest dyskietka czy też napęd, należy znaczną część badań poświęcić wpływowi temperatury na proces zapamiętywania danych. Praktyka wskazuje, że błędy pojawiają się w sytuacji, gdy jest rozregulowany napęd. Stąd badania dyskietek przeprowadzono w napędzie, który był uprzednio dokładnie wyregulowany. Testy przeprowadzano na dyskietkach tych samych producentów, co przy badaniu własności mechanicznych, a ich wyniki opisane w poprzednim zeszycie, zestawiono w tabeli 2.

W badaniach użyto jako urządzenia pomiarowego mikrokomputera Sirius 1, wyposażonego w dwa napędy dyskietkowe typu Tandon TM 100. W identyczne napędy, tylko w wersji 40-ścieżkowej, jest wyposażony IBM PC. Regulację obu napędów wykonano za pomocą dyskietki wzorcowej Dysan 206-10. Jeden z nich był idealnie wyregulowany, natomiast drugi znajdował się na granicy rozregulowania (70% zgodności obu sygnałów testujących). Dzięki metodzie zapisu GCR w komputerze Sirius można uzyskać znacznie większą gęstość zapisu niż w mikrokomputerach innych typów, a więc można było oczekiwać większego natężenia błędów.

Do celów kontroli dyskietek opracowano specjalny program, który wszystkie bajty danych na ścieżkach 1, 31, 32, 33 i 79 zapisywał wartością F0, a następnie je odczytywał. F0 wybrano dlatego, że w tym wypadku występują zawsze na przemian zera i jedynki (patrz kod GCR w tabeli 1).

Przeprowadzono następujące testy:

- zmierzono wrażliwość warstwy tlenku żelaza na podstawie poziomu odczytu wytworzonego przez sygnał testujący na ścieżce 32. W tabeli 2 podano wartości górne napięć wyjściowych wzmacniacza odczytu, które ustalono za pomocą oscyloskopu HAMEG 412;
- przeprowadzono trzy testy mające na celu ustalenie wpływu temperatury na dyskietki. Dwa z nich symulowały sytuacje, jakie mogą wystąpić podczas eksploatacji mikrokomputera. Trzeci test miał stwierdzić zachowanie się dyskietek przy przekroczeniu wskazanych przez producentów temperatur granicznych.

W pierwszym teście wszystkie 12 dyskietek zapisano sygnałem testującym w uprzednio wyregulowanym napędzie, a następnie umieszczono w chłodziarce w temperaturze 10°C i przechowano przez 18 godzin. Warunki te stanowią symulację transportu dyskietek samochodem w czasie zimy. Dyskietki kolejno wyjmowano z chłodziarki i

Tabela 1. Konwersja kodu GCR

Bity danych	Kod GCR
0000	01010
0001	01011
0010	10010
0011	10011
0100	01110
0101	01111
0110	10110
0111	10111
1000	01001
1001	11001
1010	11010
1011	11011
1100	01101
1101	11101
1110	11110
1111	10101

Tabela 2. Poziom sygnału wyjściowego wzmacniacza odczytu

Symbol wyrobu	Napięcie (mV)
BASF 73041	400
Döbbelin + Böder Disky 3222	380
3M 746-0	380
Dysan 204/2D	390
Elephant's Memory No. 8	420
Fuji MD 2D 96	390
Maxell MD-2DD	370
Memorex 1D-80	400
Rohone Poulenc Systems MN 1-DD	370
TDK M2DX-S	380
Verbatim 577-01	380
Xidex 5022-2000	440

umieszczano w wyregulowanym napędzie w temperaturze ok. 23°C, gdzie zostały one dwukrotnie odczytane, a następnie na nowo dwukrotnie zapisane.

W kolejnym teście temperaturowym postępowanie było podobne, jednak z tą różnicą, że dyskietki podgrzewano przez 5 godzin w cieplarce w temperaturze 50°C. Warunki mogą wystąpić wtedy, gdy dyskietkę położy się przez nieuwagę w pełnym słońcu lub na urządzeniu grzewczym.

Trzeci test nie dotyczył już dopuszczalnych temperatur eksploatacji, lecz składowania, które producenci określają w granicach od 4 – 50°C. Aby stwierdzić, czy zapisane na dyskietkach informacje zostaną zachowane również w warunkach bardzo niskich temperatur, a same dyskietki zachowują swe własności mechaniczne, próbki umieszczono na dwie godziny w temperaturze -20°C.

Niestety nie przeprowadzono testu żywotności dyskietek, ponieważ nie dysponowano ani wystarczającą liczbą napędów, ani niezbędnym do tego czasem badań. Firma Dysan, która zdaniem autora podaje w swych materiałach najbardziej wiarygodne dane, określa żywotność dyskietki na 200 godzin ciągłego zapisu i odczytu tej samej ścieżki. Odpowiada to ok. 4 mln obrotów dyskietki w napędzie oraz nieprzerwanej pracy przez ponad 8 dni.

Wyniki

Przy pomiarze poziomu sygnału wyjściowego okazało się, że dyskietki różnych firm miały rozrzut w granicach ok. 20%. Dyskietki firm Xidex oraz Elephant's Memory dawały najsilniejszy sygnał odczytu. Jednakże różnice były tak małe, że w nowych i dokładnie wyregulowanych napędach nie miało to żadnego znaczenia. Natomiast w starszych napędach, których głowice wytwarzają słabszy sygnał wyjściowy, należy używać dyskietek o lepszym poziomie sygnału.

Uzyskane wyniki testu dotyczącego wpływu temperatury były zaskakujące. Zarówno chłodziarka jak i cieplarka zupełnie nie wpłynęły na obniżenie jakości dyskietek. We wszystkich dyskietkach sygnał testowy został odczytany i na nowo zapisany bezbłędnie. Wynik ten był tak zdumiewający, że badania wielokrotnie powtarzano. Szczególnie zadziwił brutalny końcowy test w chłodziarce, który nie wykazał obniżki jakości dyskietek. Jedynie dyskietka firmy Xidex początkowo wykazała błąd, który jednak całkowicie zniknął po ponownym jej włożeniu do napędu. Ekstremalne warunki temperaturowe spowodowały jednak deformację osłon ochronnych niektórych dyskietek. Największe wystąpiły w dyskietkach firmy Döbbelin + Böder w wyniku podgrzewania. Deformacja spowodowała również znaczny wzrost szmerów wirowania. To ostatnie dotyczy zwłaszcza dyskietki firmy Elephant's Memory. Dla jakości zapamiętywania danych nie miało to jednak żadnego znaczenia.

Na podstawie przeprowadzonych badań wyrobów dwunastu producentów, można stwierdzić, że nie ma wśród nich wyraźnych liderów, wszystkie dyskietki bezbłędnie zapamiętują dane. Jeżeli akcent położymy na własności mechaniczne (patrz pierwsza część badań) oraz przypiszemy szczególną wagę występowaniu maksymalnie wysokiego poziomu sygnału, to na czoło wysuną się dyskietki firm Elephant's Memory, Dysan, Fuji oraz Xidex. Należy wreszcie stwierdzić, że jakość napędu ma większy wpływ na zapamiętywanie danych niż marka dyskietki. Oczywiście nie powinno to wpływać na zakup komputera...

micro

PEPSY – pakiet programowy do obliczeń statycznych

(dokończenie art. ze str. 5)

matów obciążeń. Czas ten osiągnięto na mikrokomputerze IBM PC wyposażonym w koprocesor arytmetyczny i RAM-dysk.

Krótkie czasy obliczeń

W tabeli przytoczono czasy obliczeń dla pięciu różnych przykładów z praktyki inżynierskiej. Czasy te dla różnych konfiguracji sprzętowych (konfiguracje bardziej rozbudowane są oczywiście droższe) liczone były od momentu wprowadzenia danych o strukturze i obciążeniach do chwili wyprowadzenia wyników końcowych: naprężeń w węzłach, przemieszczeń węzłów i reakcji podpór.

W przykładach rozpatrywano następujące problemy: podpory pod rurociągi w elektrowni A, dwa mosty kratownicowe B i C, rurociąg ciepłowniczy D i żuraw budowlany E. W tabeli przytoczono dane charakteryzujące złożoność zadań: liczba węzłów **wz**, liczba elementów **el**, liczba sił **po**, liczba podpór **wb**, liczba stopni swobody **ss**.

Czasy obliczeń sprawdzono na komputerze IBM PC dla trzech konfiguracji. W pierwszym wariancie użyto tylko procesora 8088. w wariancie drugim procesora 8088 wraz z koprocesorem arytmetycznym 8087. W obu wypadkach wymagane były częste dostępy do zewnętrznej pamięci masowej. Po dodatkowym zwiększeniu pamięci RAM o 384 KB i zainstalowaniu odpowiedniego oprogramowania można było zasymulować w pamięci RAM dysk o pojemności 360 KB (tzn. RAM-dysk). Był to wariant trzeci. Zależność czasu obliczeń (liczonego w minutach) od poszczególnych konfiguracji dla każdego z pięciu zadań przedstawiono graficznie na rys. 4.

Jak widać z tych przykładów, program PEPSY umożliwia niewielkim kosztem znaczne odciążenie inżyniera przy wykonywaniu pracochłonnych obliczeń statycznych.

micro

BIT ZA TRZY GROSZE

Spektroskopia Spectrum

Proponujemy wprowadzenie nowej dziedziny wiedzy. Jest nia spektroskopia informatyczna. Nauka ta zajmuje sie badaniem widm programow komputerowych i okreslaniem na ich podstawie jakosci, stopnia komplikacji i zawartosci badanego programu.

Kazdy program sklada sie jak wiadomo z kolejnych slow. Dla osmiobitowego procesora slowa te to liczby z zakresu od 0 do 255.

Mozna wiec zadac sobie pytanie, ile razy dane slowo wystepuje w programie i jakie sa proporcje czestosci wystepowania roznych slow. Program zamieszczony ponizej pozwala na wykonanie analizy spektroskopowej za pomoca histogramu wystepowania roznych slow w interesujacym nas obszarze pamieci. Po wprowadzeniu programu i uruchomieniu go instrukcja RUN (po ewentualnym wprowadzeniu badanego bloku programu) wystarczy podac adres poczatkowy i koncowy badanego obszaru, a na ekranie dostaniemy wykres przedstawiajacy widmo pamieci.

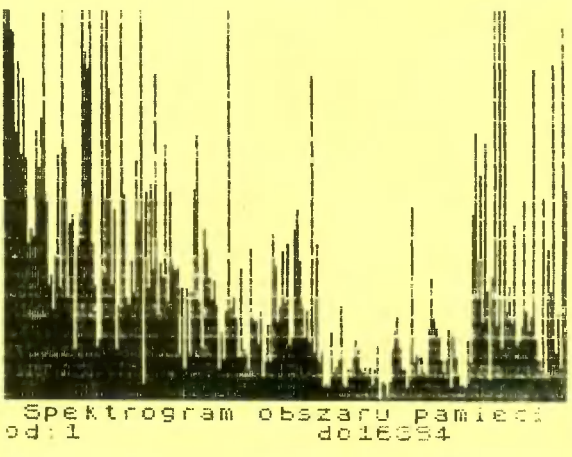
Jako przyklad prezentujemy widmo pamieci ROM w ZX Spectrum. Widac wyraznie potrojna linie ROM o dlugosciach 201,203,205 (rozkazy RET CALL i liczba 203 poprzedzajaca rozkazy dwubajtowe). Silne linie o malych dlugosciach sugeruja czeste uzywanie rachunkow, ktorych rozkazy maja male wartosci.

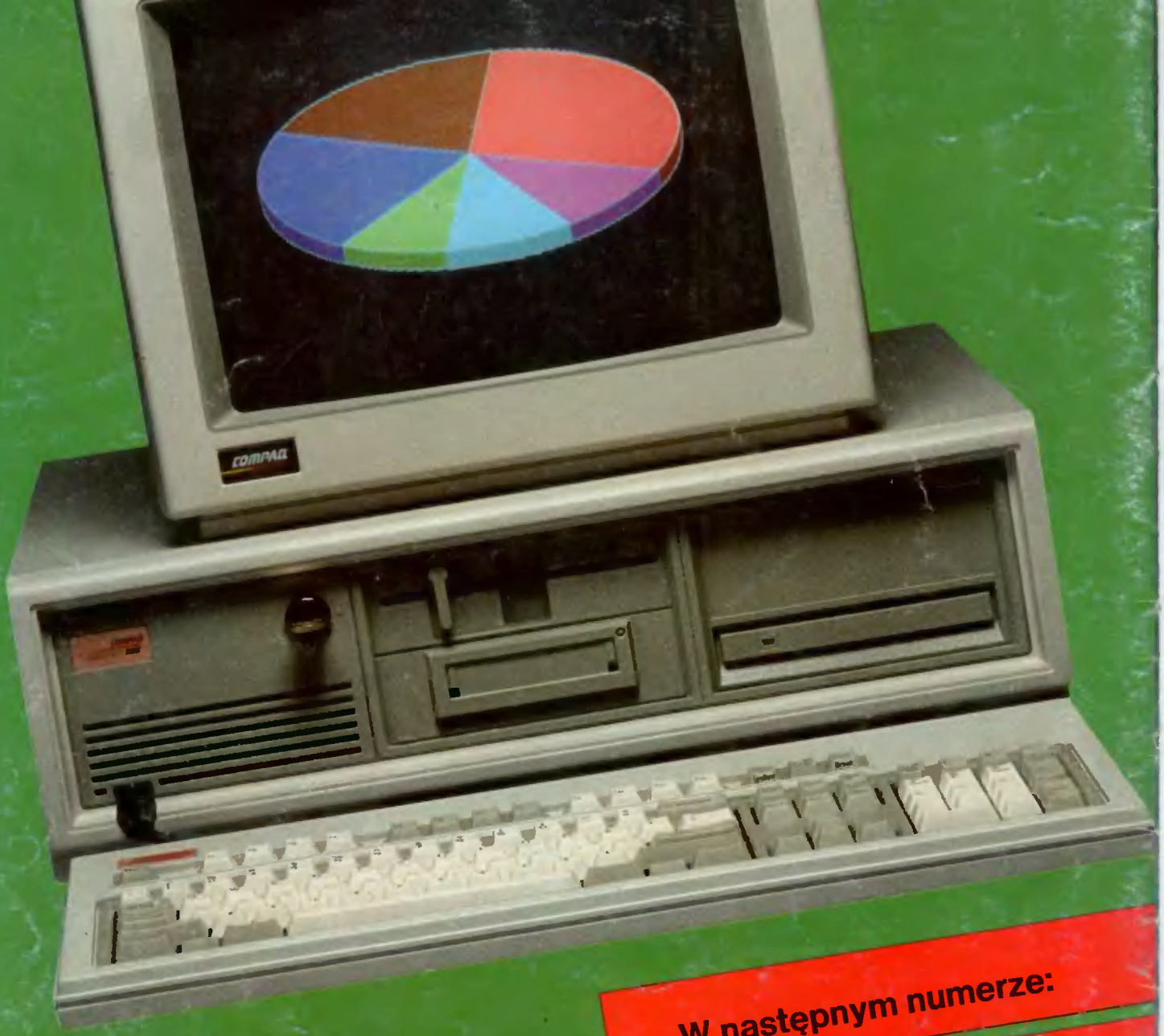
Mozna zauwazyc pewne prawidlowosci w widmach roznych programow. I tak gry typu ADVENTURE maja silny pik dla liczb o wartosciach w okolicy 100 (litery alfabetu). W grach prostych, w ktorych wystepuje duzo grafiki a malo obliczen, rozklad jest prawie rownomierny. Natomiast w programach uzytkowych (np. kompilatory) wyraznie przewaza "promieniowanie krotkofalowe".

Ciekawe jest, ze trudno znalezc program, w ktorym linia 0 miesci sie na ekranie. "0" nie jest uzywane w Basicu, w programach kodowych oznacza NOP, a w grafice powoduje nadruk pustego bajtu. Czyzby w programach bylo tak duzo "niepotrzebnych" komorek?

BARTEK MANK

```
  1 REM Bartek Mank 1986
 10 CLEAR 31999
 20 PRINT AT 10,0;"Jesli chcesz
 zaladowac badany blok danych -
wcisnij klaiwsz T"
 30 IF INKEY$="" THEN GO TO 30
 40 IF INKEY$="t" OR INKEY$="T"
 THEN PRINT : PRINT TAB 10; FLAS
H 1;"Wlacz magnetofon";: LOAD ""
CODE 32000: CLS
 50 RESTORE : FOR i=64000 TO 64
052: READ a: POKE i,a: NEXT i
 60 INPUT "poczatek badanego bl
oku (>0)";a: IF a=0 THEN LET a=1
 70 INPUT "koniec badanego blok
u (>0)";a1: IF a1=0 THEN LET a1=
1
 90 RANDOMIZE 1: POKE 64053,PEE
K 23670: POKE 64054,PEEK 23671
 100 RANDOMIZE a1: POKE 64055,PE
EK 23670: POKE 64056,PEEK 23671
 110 CLS : PRINT AT 20,0;" Spekt
rogram obszaru pamieci ","od:";a
,"do";a1
 120 RANDOMIZE USR 64000
 130 DATA 33,57,250,54,16,229,20
9,19,1,0,1,237,176,42,53,250,229
,126,33,57,250,79,6,0,175,237,74
,126,254,175
 135 DATA 40,2,60,119,71,205,229
,34,225,35,237,91,55,250,124,186
,32,224,125,187,32,220,201
```





W następnym numerze:

o komputerach

Amstrad Joyce i PC 1512

mikroklan 2'87 KUPON

Jeśli chcesz wylosować wyposażenie
lub sprzęt mikrokomputerowy,
wytnij oraz zachowaj kupon
i oczekuj następnego numeru czasopisma
(szczegóły na str. 21).